American Journal Of Applied Science And Technology
(ISSN – 2771-2745)
VOLUME 04 ISSUE 10 Pages: 69-79
OCLC – 1121105677
Crossref doi Google Scholar WorldCat MENDELEY

Publisher: Oscar Publishing Services

Journal Website: https://theusajournals.com/index.php/ajast



Research Article

# DYNAMIC PACKET FILTERING USING MACHINE LEARNING METHODS

Submission Date: October 13, 2024, Accepted Date: October 18, 2024,
Published Date: October 23, 2024
Crossref doi: https://doi.org/10.37547/ajast/Volume04Issue10-11

**Sarvar Norboboyevich Tashev**
**Shakhrisabz Branch of Tashkent Chemical-Technological Institute, Uzbekistan**

## ABSTRACT

With the emergence of the Internet, cyber-attacks and threats have become significant issues. Traditional manual network monitoring and rule-based packet filtering methods have become labor-intensive and less effective in combating attacks. Filtering packets based solely on payload and pattern matching is also inefficient. There is a need for a dynamic model capable of learning packet filtering rules. This article proposes a packet filtering model using Neural Networks. After developing the model classified with training and validation data, it can be utilized to support dynamic packet filtering. The proposed model allows filtering packets not only based on static rules but also considering IP packet attributes and rules learned by the model in advance. The model takes into account payloads and other IP packet attributes for filtering. It can automatically update firewall rules to enhance security.

## KEYWORDS

Cyber-attacks, Neural Networks, IP packets, firewall.

## INTRODUCTION

People have become highly dependent on networks for daily tasks. Computer networks are designed to handle high traffic demands and meet real-time constraints. Many technologies and components are involved in transmitting or filtering packets from one network device to another, with much of the process traditionally managed manually. As a result, network security issues have become significant because packets or data pass through multiple components to reach their destination. Modern network applications must support network functions while network

American Journal Of Applied Science And Technology
(ISSN – 2771-2745)
VOLUME 04 ISSUE 10 Pages: 69-79
OCLC – 1121105677
Crossref doi Google Scholar WorldCat MENDELEY
Publisher: Oscar Publishing Services

functionalities are virtualized and control is increasingly handled through software.

Network applications help operators manage and monitor traffic, and improve network systems by analyzing data. Numerous studies in the field of networking have opened up opportunities for autonomous networking applications, with network security and packet filtering being one of the most important applications.

This paper proposes a dynamic model that can learn packet filtering rules. Neural Networks (NNs) are used for the model, with a comparison made to Support Vector Machines (SVM). The model can classify packets that were not included during the training phase. The proposed model enables packet filtering not only based on static rules but also considering IP packet attributes, allowing for the classification of legitimate and malicious packets. Additionally, the model can automatically update firewall rules in the IP tables.

The NN model was implemented using Python and the Keras framework. The long-term goal is to improve the efficiency of packet filtering. Two additional features were introduced to enhance model accuracy:

1. Device-based packet filtering
2. Subnet-based packet filtering

**A. Packet Filtering**

Packet filtering allows a network operator to perform actions (permit or block) based on the following factors:

- Source address of the data
- Protocols used for transmission, such as transport and/or application layer protocols

Most packet filtering systems do not base their decisions on the content of the data, meaning they do not make content-based decisions. Packet filtering provides a certain level of protection for the entire network when placed at strategic locations such as gateway routers or edge devices, serving as a shield for the network. This makes packet filtering crucial for network security, regardless of the website's size, and it is implemented transparently for end-users. Unlike proxying, packet filtering does not require any special software or configuration on the user machines, nor does it need any special training or procedures for the users.

**1.1 Supervised Machine Learning**

There are three main categories of Machine Learning (ML) techniques: unsupervised learning, semi-supervised learning, and supervised learning. The primary goal of unsupervised learning techniques is to identify patterns, structures, or knowledge in unlabeled data. Semi-supervised learning occurs when part of the data is labeled either during data collection or by human experts. Labeled data plays a crucial role in solving the problem. If the data is entirely labeled, the technique is referred to as supervised learning.

Most supervised ML techniques follow training, validation, and testing phases. Labeled data consisting of the necessary attributes for packet filtering is used for training, making supervised ML algorithms applicable. The labeled features typically represent business or problem variables deemed important by experts for validating the data and testing the ML model.

The proposed approach aims to address challenges associated with manual network monitoring and rule-based packet filtering, making the system more

American Journal Of Applied Science And Technology
(ISSN – 2771-2745)
VOLUME 04 ISSUE 10 Pages: 69-79
OCLC – 1121105677
Crossref doi Google Scholar WorldCat MENDELEY
Publisher: Oscar Publishing Services

adaptable to evolving threats and reducing human intervention.

### 1.2 Existing Packet Filtering Methods

Various techniques have been proposed for defining firewall policies and mechanisms to verify filtering rules, which help reduce dependencies. Researchers have evaluated the performance of firewalls in distributed systems in terms of transaction time and latency. In [3], the authors developed a functional model of firewalls, including an algebraic representation for describing access rules and a formal tool for configuring the firewall. The approach also incorporates automatic anomaly detection for the insertion and definition of rules.

In [4], a tool was introduced for writing and modifying firewall rules. Firewalls function as logical separators, restrictors, and analyzers, and their physical implementation can differ across locations [5]. Typically, a firewall consists of a combination of physical components such as routers, host computers, or a mixture of routers, computers, and networks, along with the necessary software [6]. The configuration of the firewall depends on the security policy, budget, and the overall functioning of the object [7].

Deri described dynamic packet filtering using the Counting Bloom Filter (CBF) [8]. Although CBF addresses issues related to adding and removing components and offers improvements over previous static filtering processes, it still has limitations, such as low memory utilization, limited rule capacity, and a high false positive rate [9]. Modern applications like Voice over IP (VoIP) and peer-to-peer (P2P) traffic monitoring require dynamic packet filtering based on attributes beyond simple VLAN/IP address/port number characteristics. Common packet filtering techniques such as the Berkeley Packet Filter (BPF) [9] and router-based Application-Specific Integrated Circuits (ASIC) filtering are not sufficient for these applications.

Abeni et al. [10] proposed a solution for rapid and compact packet filtering based on partitioning rule databases and storing them in fast and compact Bloom filters. A specialized clustering technique is used for database partitioning, and the results showed that even a large set of rules can be reduced to a minimal number of segments and stored in compact Bloom filters.

Overall, traditional packet filtering methods have various shortcomings that need addressing, especially in dynamic and high-performance network environments. Modern approaches seek to overcome these limitations by leveraging advanced data structures, machine learning techniques, and adaptive rule management strategies to enhance filtering efficiency and network security.

## METHODOLOGY

The proposed approach for dynamic packet filtering is described as follows:

1. Packet Sniffer: A packet sniffer is developed using Python's socket library. This tool captures network packets in real-time. The captured data is then stored in CSV format using the Pandas library, allowing for easy manipulation and analysis of the packet information.

2. Data Preprocessing: The collected data undergoes preprocessing to convert categorical information into numerical format using encoding techniques. This step ensures that the data is suitable for machine learning models, which typically require numerical inputs.

American Journal Of Applied Science And Technology
(ISSN – 2771-2745)
VOLUME 04 ISSUE 10 Pages: 69-79
OCLC – 1121105677
Crossref doi Google Scholar WorldCat MENDELEY
Publisher: Oscar Publishing Services

3. Model Construction: The preprocessed data is used to train a neural network (NN) model. This model is designed to learn from the patterns in the data and effectively classify packets based on their attributes.

4. Model Application: Once the NN model is trained, it is deployed for practical use. The implemented model has the capability to dynamically filter packets with high accuracy, distinguishing between legitimate and potentially harmful packets based on learned patterns.

This methodology aims to enhance the effectiveness of packet filtering by leveraging machine learning techniques, allowing for adaptive responses to evolving network threats and improving overall network security.

**1. Diagram for Dynamic Packet Filtering Architecture**

Packet Sniffing

A packet sniffer is a device or application that monitors network traffic. It is also known as a packet analyzer or network analyzer. These packets have their own addresses and are intended for specific devices. Sniffers can be constructed in two ways:

- Unfiltered Sniffer: This captures all available packets on the network.
- Filtered Sniffer: This allows analysts to collect packets that contain only selected data components【11】.

Packet sniffing has various applications and is commonly used for troubleshooting network issues. It can identify misrouted packets or packets that should not be present on the network. Unintended packets for specific ports may indicate misconfiguration of one or more nodes. A specialized data sniffing algorithm is designed to collect and exchange packets on the network, train the neural network (NN) model, and identify malicious packets.

**Data Preprocessing**

In our approach, the preprocessed data is obtained in two stages: data processing and normalization.

**2.1 Data Processing**

Raw data (in text, image, or video format) needs to be converted into a suitable format for the machine learning (ML) model to ensure high-quality data preparation before applying ML techniques【12】. This stage involves deleting incorrect, incomplete, and inaccurate data, as well as replacing missing values. This step checks the usability, confidentiality, and integrity of the data. According to our approach, the following methods have been accepted for data processing:

- Smoothing: This method helps to eliminate some noise in the dataset, thereby supporting the identification of essential features.
- Aggregation: This method is used to combine related data. This stage is critical, as the accuracy of data depends on the quantity and quality of the information.
- Discretization: This method is used to divide continuous data into intervals. Discretization reduces the size of the data.
- Converting Categorical Data to Numerical Data: For example, converting IP addresses to numerical data enhances accuracy and efficiency. For instance, "127.0.0.1" is transformed to 127001.

This architecture serves as a foundation for dynamic packet filtering by utilizing a structured approach to

American Journal Of Applied Science And Technology
(ISSN – 2771-2745)
VOLUME 04 ISSUE 10 Pages: 69-79
OCLC – 1121105677
Crossref doi Google Scholar WorldCat MENDELEY
Publisher: Oscar Publishing Services

sniffing and processing network packets, ultimately improving the identification and filtering of malicious packets.

### 2.2 Normalization

Normalization is the process of rescaling or transforming initial data to ensure that each feature has an equal influence. This process addresses significant data issues, such as dominant features and outlier values, which can affect the performance of machine learning (ML) algorithms during training【13】.

In this research, the process of converting data to a specified range (e.g., between 0 and 1 or between -1 and 1) is utilized. The minimum and maximum values of the unnormalized data are applied to normalize the data. This method ensures that the unnormalized data conforms to a linear range of upper and lower bounds. Typically, data is rescaled to fit within a range of either 0 to 1 or -1 to 1.

Data quality is crucial for training and predicting with the model. Table I presents some sample packets collected in real time by the packet sniffer, which are then preprocessed (through smoothing, aggregation, and discretization) to adapt them to the proposed model. Only the attributes necessary for inputting into the neural network (NN) model are filtered. All fields with numerical values remain unchanged. Column E of Table I shows the two possible actions for each packet: allowing or blocking it for model training. The concept of restrictions is illustrated using known malicious sources and certain websites established on the server for other reasons, meaning that packets matching these attributes or originating from these sites are blocked. For instance, for prototyping, we consider malicious packets from specific YouTube or Twitter channels. Additionally, some data that leads to the detection of malicious software is accepted for the model. Consequently, such packets are used in training to prevent their re-entry into the system, which means the action for these packets will be blocking. Data is crucial for prediction, allowing the model to understand which types of packets to permit or block.

Neural networks (NN) are a type of supervised machine learning that utilizes labeled data. The NN model is capable of analyzing complex and highly uncertain relationships between input and output variables, and it has been demonstrated that a network with only one hidden layer but sufficient nodes can represent any functionality. A simple NN architecture consists of three layers: the input layer, the hidden layer, and the output layer【14】. Each layer is composed of nodes or neurons. Weights in the NN represent the factors or coefficients that indicate the magnitudes of the connections between neurons in adjacent layers. Important features are identified by retraining the NN to minimize the loss function between predicted and actual results.

**Keras Framework**

Keras is a high-level Python library for neural networks (NNs) that operates as a module for supervised machine learning algorithms. NNs possess self-learning and self-adaptive properties, making them suitable for addressing some complex uncertain prediction problems. Our approach utilizes Keras, which can be based on either TensorFlow or Theano【15】. Keras includes the following modules: activation functions, layer modules, preprocessing modules, objective function modules, and optimization technique selection modules, among others. These modules facilitate the straightforward creation of network models and the tuning of essential parameters within the NN.

American Journal Of Applied Science And Technology
(ISSN – 2771-2745)
VOLUME 04 ISSUE 10 Pages: 69-79
OCLC – 1121105677
Crossref doi Google Scholar WorldCat MENDELEY
Publisher: Oscar Publishing Services

### Activation Functions

In the proposed model, two types of activation functions are utilized: ReLU and Softmax.

### ReLU (Rectified Linear Unit)

ReLU is a widely used non-linear activation function in neural networks. Mathematically, it is defined as F(x)=max⁡(0,x)F(x) = \max(0, x)F(x)=max(0,x), where xxx is the input to the neuron. The advantage of ReLU is that all neuron cells may not activate simultaneously, meaning if the result of a linear transformation is zero, the neuron becomes inactive. As a result, ReLU is more efficient compared to many other functions, as a small number of neurons are active at any given time. If the gradient value is zero, the weights adjusted during the backpropagation phase of NN training remain unchanged【16】.

### Softmax

The Softmax function is a multi-dimensional generalization of the logistic function and is a variant of the sigmoid function that is often used in conjunction with other functions. The sigmoid function generates values between 0 and 1, allowing for interpretation as the probability of a sample data point belonging to a particular class. Unlike the sigmoid function used for binary classifiers, the Softmax function can solve multi-label classification problems. It returns the probabilities for different classes for each data point【16】.

In this research, only two outputs exist: allow or block. Therefore, using the ReLU activation function in the output layer is the best option. If we were to create a network or model for multi-class classification, the output layer of the NN would contain a number of neurons corresponding to the target classes.

### Neural Network Model Design

The neural network (NN) is constructed with an input layer and four additional layers. Choosing the number of nodes in the input layer is a challenging task that usually requires the developer's knowledge and several trials. If there are too many units in the hidden layers, the training time becomes excessively long, the error may not be optimal, and the phenomenon of "overfitting" may occur. Therefore, experimental evaluation may also be necessary to determine the number of hidden layer elements for efficiency.

### Input Layer

The input layer consists of six nodes, representing the number of data inputs to the NN. These inputs include the following attributes: source IP address, source port number, destination port number, acknowledgment number, sequence number, and data received with the packet. Unlike many anomaly detection methods, the proposed model does not utilize the destination IP address, as the detection program is working on the destination machine for identification purposes rather than for other devices on the network. The proposed model operates on the server to detect potential anomalies. The activation function used in the input layer is Softmax.

### Hidden Layers

The NN model comprises three hidden layers:

1. First Hidden Layer: This layer has 64 nodes. The six input values are mapped to these 64 values using the ReLU activation function.

2. Second Hidden Layer: This layer has 256 nodes. The 64 values obtained from the first hidden layer are also mapped to these 256 nodes using ReLU.

American Journal Of Applied Science And Technology
(ISSN – 2771-2745)
VOLUME 04 ISSUE 10 Pages: 69-79
OCLC – 1121105677
Crossref doi Google Scholar WorldCat MENDELEY
Publisher: Oscar Publishing Services

3. Third Hidden Layer: This layer has 512 nodes. The 256 values obtained from the second hidden layer are mapped to these 512 nodes using ReLU.

**Output Layer**

The output layer consists of two nodes, representing the target attribute; hence the output will be either 1 or 0. If the packet is malicious, "1" is activated; if the packet is legitimate, "0" is activated. The two values are mapped from the 512 values of the third hidden layer using the ReLU activation function.

The training dataset is collected from internet sources and preprocessed by removing all unnecessary attributes and noise to adapt it to our model. It is then used in the training and validation processes to evaluate the model's accuracy. The dataset is divided into two parts: training data and validation data.

In this study, 80% of the total collected and preprocessed dataset is used for training and validation of the neural networks (NN), while 20% is used for testing the model. Keras is utilized to create the sequential NN. Similar to many other aspects of machine learning, the data splitting ratio is determined considering the target problem, the network, and all relevant characteristics of the dataset.

This sequential model contains four layers: three hidden layers and one output layer. The input layer consists of six nodes, with one node assigned for each attribute. The activation function used in the input section is ReLU. The values obtained from the third hidden layer are passed to the output layer, which contains two nodes indicating whether the packet is legitimate or malicious. These input layers are transformed into binary values (0 and 1) using the Softmax activation function in the output layer. Thus, as a result of processing data through all layers, six inputs correspond to two values. The accuracy obtained during training and testing with 30 epochs is equal to 100%.

**Comparison with Support Vector Machine (SVM)**

For comparison, a second model based on Support Vector Machine (SVM) was developed. The accuracy obtained using SVM for training was approximately 81.1%, and for testing, it was about 81%. When comparing these two models (SVM and NN), the NN demonstrates significantly higher accuracy, with a training loss of approximately 1.2781×10−51.2781 \times 10^{-5}1.2781×10−5 and a testing loss of 1.1917×10−51.1917 \times 10^{-5}1.1917×10−5. Consequently, the final model for dynamic packet filtering was developed based on the NN architecture.

## RESULTS

### A. Packet Filtering

Packet filtering is supported by firewall techniques to monitor both internal and external networks, and it is implemented by detecting actions (allow or block) for packets based on inspection and selected attributes. If the model's result is not harmful, the packet is considered safe and verified. High-level protocols and their associated attributes, such as User Datagram Protocol (UDP) and Transmission Control Protocol (TCP), can also be taken into account. Figure 1 shows the sample prediction results for packets. The predictions were evaluated based on the following criteria: size of the training dataset, NN design, number of iterations, and accuracy. Additionally, we expanded the implementation with two additional features:

1. Device-based packet filtering
2. Subnet-based packet filtering

American Journal Of Applied Science And Technology
(ISSN – 2771-2745)
VOLUME 04 ISSUE 10 Pages: 69-79
OCLC – 1121105677
Crossref doi Google Scholar WorldCat MENDELEY
Publisher: Oscar Publishing Services

**B. Device-based Packet Filtering**

We used the same ML model to establish device-based packet filtering since each digital device has a unique identifier and address known as a MAC address. MAC filtering is a security solution based on access control. The MAC address assigned to each device is a 48-bit address used to evaluate whether it has permission to connect to the network.

The main action of the device-based packet filtering technique is to identify the targeted malicious MAC address and enhance protection against harmful packets by adding it to automatic firewall rules. This is accomplished by comparing the MAC address to a pre-defined list of addresses. The MAC address of the device is obtained through Address Resolution Protocol (ARP) table acquisition and is automatically added to the firewall, where the source and destination IP addresses of harmful packet flows are identified along with the device's MAC address. Acquiring the ARP table implies mapping network addresses to MAC addresses, where the ML model retrieves information regarding the device's MAC address. Once a harmful packet is identified, filtering is applied, and the MAC address is added to the automatic firewall rules, ensuring continuous protection from harmful packets.

The following algorithmic process is used:

```
f = open("FilteringRules.txt", "a")
if "ether" in data or "at" in data:
    malMac = data.split("at")[1].split(" [ether]")[0]
    ruleMAC = "iptables -I INPUT -s " + str(s_addr) + " -p tcp --dport " + str(dest_port)
+ " -m mac --mac-source " + str(malMac) + " -j DROP"
    os.popen(ruleMAC)
    f.write(ruleMAC + "\n")
if malMac == "":
    ruleIP = "iptables -I INPUT -s " + str(s_addr) + " -p tcp --dport " + str(dest_port) +
" -j DROP"
    os.popen(ruleIP)
    f.write(ruleIP + "\n")
f.close()
```

Figure 3(a) displays the results after executing our algorithm, showing that the harmful packets were successfully detected by the model. Thus, after the firewall was updated, connection requests were denied access to the device. The acquisition of the MAC address and its submission to the firewall rule occurs automatically in the background, as depicted in Figure 3(b), which compares the firewall and modifications, indicating that the MAC address has been updated in the firewall section. After detection and filtering are applied, harmful packets, as previously mentioned, do not enter this device. The instantaneous screenshots are provided in Figure 3, showing (a) the results of device-based packet filtering and (b) the updates to the MAC address in the firewall section.

(a)

```
C: /Desktop/packet# echo “hello” | rc -cu localhost 8080
Localhost [127.0.0.1] (?) : Connection refused
C: /Desktop/packet#
```

American Journal Of Applied Science And Technology
(ISSN – 2771-2745)
VOLUME 04 ISSUE 10 Pages: 69-79
OCLC – 1121105677
Crossref doi Google Scholar WorldCat MENDELEY
Publisher: Oscar Publishing Services

(b)

```
C: /Desktop/packet# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -s 127.0.0.1/32 -p tcp -m tcp --dport 34860 -m mac --mac-source 00:50:56:e0:f6:fd
-j DROP
-A INPUT -s 127.0.0.1/32 -p tcp -m tcp --dport 8080 -m mac --mac-source 00:50:56:e0:f6:fd
-j DROP
C: /Desktop/packet#
```

### C. Subnet-based Packet Filtering

In the initial approach, filtering was based on individual packets, using various attributes of the packets to determine their safety. Expanding this initial approach, a logical way to support driver access to the network or subnet becomes evident. Allowing a driver to enter the network can lead to inefficiencies in packet-based filtering. Thus, MAC address-based filtering is employed, as devices within a subnet are associated via their MAC addresses. By combining these two scenarios, the system is protected from both external threats and hackers within the network.

This additional development entails blocking the driver from the network or subnet because well-monitored environments constantly prevent hackers from entering the system. In this scenario, an algorithm is employed to secure the driver's safe packets, identifying the IP address from IPv4 packets and implementing the IP table rules derived from the packets. This method is designed to block all packets from subnets identified against the target. Each task of this process is programmed, allowing for automatic detection and implementation of IP table rules.

The scripting code illustrates the method for filtering packets based on subnet:

```
subnet = os.popen("""ifconfig | grep "inet " | grep -v "127.0.0.1" | awk '{print $2}'""")
# 127.0.0.1 is excluded, as it is the address where the program is running.
subnet = subnet.read()
subnet = subnet.split(".")
subnet = subnet[0] + "." + subnet[1]
if result == 0:
    continue
else:
    if s_addr.startswith(subnet):
        continue
    else:
        subnetBlock = str(s_addr).split(".")
        subnetBlock = subnetBlock[0] + "." + subnetBlock[1] + ".0.0/16"
        os.popen("iptables -A INPUT -s" + subnetBlock + " -j DROP")
        print("Blocking the subnet location: " + subnetBlock)
```

If the NN predicts that a packet is not harmful, no action is taken. However, if the packet is classified as harmful, the common practice for network attacks is to set the subnet mask to 255.255.0.0. The attacker's IP address is extracted from the IP packet. Now, with both the IP address and subnet mask available, a

American Journal Of Applied Science And Technology
(ISSN – 2771-2745)
VOLUME 04 ISSUE 10 Pages: 69-79
OCLC – 1121105677
Crossref doi Google Scholar WorldCat MENDELEY
Publisher: Oscar Publishing Services

bitwise AND operation is performed between the subnet mask and IP address to determine the device's subnet. Subsequently, an IP rule is created to block all packets routed from the computed subnet. This rule is applied to the system's IP table using the OS module.

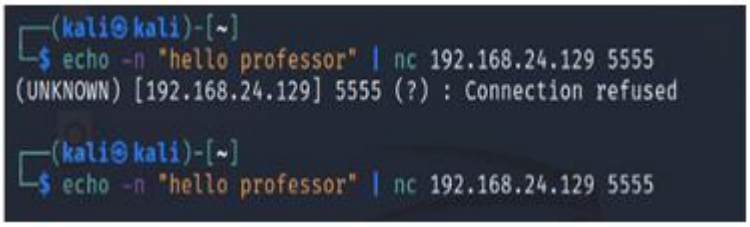


**Figure 2. Results of the source connecting the specified address**

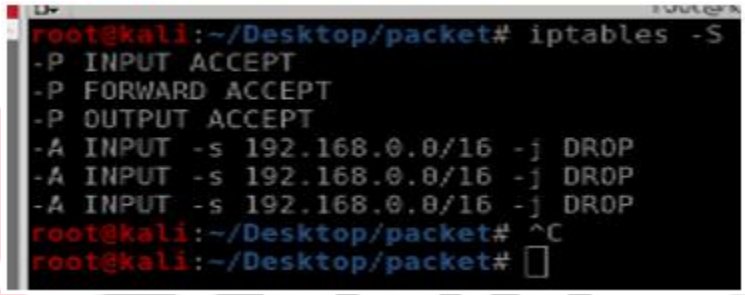


**Figure 3. Subnet IP table rule for blocking it.**

Figures 2 and 3 show partial results of the method for filtering packets by subnet. Two machines from two different networks were used for the experiment. Figure 2 shows the source attempting to send a malicious packet, but the connection is refused. Figure 3 displays the operational address of the filtering tool. Once deemed malicious, you can observe the actions taken before and after creating IP table rules and blocking the subnet. Thus, Figure 3 illustrates the creation of an IP table rule to identify and block the malicious subnet.

## CONCLUSION

This article discusses a dynamic approach to packet filtering using neural networks to support defense against attacks. This model is capable of dynamically blocking attacks. A data sniffing algorithm was employed to gather a real-time updated dataset for testing the model and analyzing its performance. We successfully implemented the dynamic packet filtering approach and introduced two additional features, namely device-based packet filtering and subnet-based packet filtering. These additional features assist in mitigating attacks from both internal and external sources. The model can be expanded to incorporate capabilities for detecting various anomalies. It is also necessary to retrain the models frequently and over extended periods. Future research should focus on incremental learning and continuous learning.

## REFERENCES

1. Gulomov Sh.R. Types of malicious traffic in the network and their detection. Multidisciplinary

American Journal Of Applied Science And Technology
(ISSN – 2771-2745)
VOLUME 04 ISSUE 10 Pages: 69-79
OCLC – 1121105677
Crossref doi Google Scholar WorldCat MENDELEY
Publisher: Oscar Publishing Services

Scientific Journal. December, Issue 24 | 2023, pp. 424-432.

2. SN Tashev, AG Ganiev The Role of "Imagination" in the Process of "Creative Thinking" Developing Students' "Imagination" and "Creative Thinking" Skills in Teaching Physics. Annals of the Romanian Society for Cell Biology, 2021/3/6, pp. 633-642.
3. SN Tashev THE ROLE OF "IMAGINATION" IN THE PROCESS OF "CREATIVE THINKING", DEVELOPING STUDENTS' "IMAGINATION" AND "CREATIVE THINKING" SKILLS IN TEACHING PHYSICS PSYCHOLOGY AND EDUCATION, pp. 3569-3575.
4. Y.B. Karamatovich, T.S. Norboboevich, N.I. Ibrohimovich. Verification of the pocket filtering based on method of verification on the model. 2019 International Conference on Information Science and Communications Technologies (ICISCT).
5. J. Ning et al., "Pine: Enabling privacy-preserving deep packet inspection on TLS with rule-hiding and fast connection establishment," in Proc. Eur. Symp. Res. Comput. Secur., 2020, pp. 3–22.