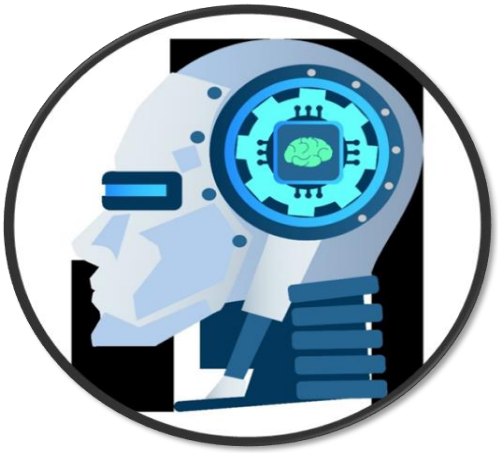
NETWORK INTRUSION DETECTION

Machine learning Algorithm



Kholoud Ali Alsoqour
King Abdul-Aziz University
Facility of Computing and information Technology

Abstract:

NIDS are critical component in protecting the networks of an organization since they can detect various invasions. The constant emergence of complex threats and the traffic load in computer networks are growing exponentially, and traditional security solutions are not enough in this case. This paper focuses on the differential use of machine learning to improve the performance of NIDSs in particular. Namely, it focuses on the analysis of Support Vector Machines (SVM) and K-means clustering algorithms. SVM is a supervised learning techniques that is very efficient in classification of high dimensions and hence plays a very big role in differentiating normal and malicious traffic. K-means which is an unsupervised learning algorithm sorts behaviors similar to the network and defines sophisticated actions as valuable by singling out odd cases as defects. This work also highlights some of the issues that are currently facing NIDS such as high traffic rate, dynamically changing threats, false positives and false negatives and encrypted traffic. Furthermore, interaction between NIDS and other security layers like firewalls, IPS, and SIEM is considered in order to describe the efficient security tactic.

Introduction

(Mukherjee, Heberlein and Levitt, 1994) The main purpose of NIDS is to identify some violation or malicious attempt to penetrate through the network which can be potentially dangerous for the security of the specified network and the data being processed in the network.

Major Parameters of Network Intrusion Detection

Traffic Monitoring:

This network is under constant surveillance by NIDS as the data packets flow through the network they are captured.

Examples of traffic capturing tools include, Wireshark, Tcpdump, and particularly sensor in a NIDS environment.

Analysis Engine:

The packet analyzer whose job is to analyze the traffic captured by NIDS with a view of detecting anomalous traffic, or traffic that has attack signatures.

This can be undertaken based on a range of methods namely; the signature based, the anomaly based and the hybrid methods.

Signature-Based Detection:

Based on a library of known threats, it looks for the signatures or patterns of the activity.

Good for the detection of known threats and threats that are in the black list but can poorly discover emerging or unknown threats.

Anomaly-Based Detection:

Uses normal network activities as a starting point and identifies activities that are beyond the norm.

Can detect new threats but can generate greater number of false positives compared to other methods such as signature-based techniques.

Alerting and Reporting:

NIDS sends out alarm notifications to system administrators when it identifies any malicious activities at the network.

Documentation is also kept as an account of the findings and a record of the occurrence.

Algorithms used for Network Intrusion Detection:

Supervised algorithm:

(Sinclair, Pierce and Matzner, 2003) Network intrusion detection systems (NIDS) are considered to be built on supervised learning algorithms because these algorithms learn normal traffic and patterns associated with the possible malicious

traffic. Here is a more detailed look at some key supervised learning algorithms used in this context:

Decision Trees:

(Song and Lu, 2015) Decision trees are learned models that are developed with the intention of modeling the decision and the possible outcomes that could occur out of it in form of a tree. Every node in the tree is equivalent to a decision that is made with reference to a particular attribute in the data and the branches depict the decision outcomes.

How It Works: The algorithm separates the data to branches that are created based on the features' value, in other words, to try to make nodes as pure as possible, meaning that the node has many data of one class. This process continues until the tree reaches to the maximum level or the nodes in the tree are sufficiently pure.

Advantages:

- Quite simple to comprehend and analyze.
- It is applicable on both numerical and nominal data types.

Disadvantages:

- Slightly sensitive to overfitting specially if the tree is deep.
- Sometimes can become really confusing if the data is massive or the features are numerous.

Random Forests:

The random forest is an ensemble technique where after training the model builds more than one decision trees and uses them in parallel and blends the results to largely enhance the performance and to minimize the overtraining.

How It Works: A new tree in the forest is created by choosing a random sample of the training data with replacement, and for split in the trees, a random subset of features is used. The last category is arrived at by averaging the scores (for regression) or through voting (for classification) as per the trees.

Advantages:

- More accurate and powerful models than that of decision tree.
- Averaging multiple trees cuts down on overfitting.
- Efficient for application on large data sets and large number of features.

Disadvantages:

- Not as efficient in terms of computational complexity than a single decision tree.
- It interprets lower than the individual decision trees.

Support Vector Machines (SVM):

(Seong, None Ha-Nam Nguyen and Sou, 2005) SVMs are a type of supervised learning models, which aim to identify the best hyperplane, that would, in turn, maximize the distance between different classes of data. In other words, the objective is to bring the classes as far apart as possible in the feature space: thus, the model is less sensitive to errors of classification.

How It Works: For SVMs, a kernel function is employed in order to map the original data space to a higher even where a hyperplane can be drawn to separate the classes. Kernels that can be used in support vector machines include linear kernel, polynomial kernel and radial basis function (RBF) kernel.

Advantages:

- Good when the number of dimensions is significantly higher than the number of samples being used.
- Implements well regarding separation of the classes of objects with a clear margin.

Disadvantages:

- Lengthy especially in scenarios with large databases.

- The choice of the kernel and tuning parameters such as the regulation parameter, kernel parameters is equally a cumbersome task.

Unsupervised learning Algorithms:

Since these kinds of algorithms do not impose the usage of labeled data, they can be used to identify new and unforeseen threats in network traffic. They operate by analyzing the data and recognizing patterns and such things as a new structure that could indicate an intrusion.

***k*-means Clustering:**

(International Journal of Reliability, Quality and Safety Engineering, 2024) k clusters, that is the data points that belong to each cluster possess the nearest mean from the rest of the clusters. K-means is one of the widely used clustering categorization that enhances the partitioning of the data into.

How It Works:

Initialize *k* centroids randomly. Place every single point of data in the context of the nearest cluster or in other words, centroid. k clusters. Reassessment of the centroid so that it represents the average relative position of the data points belonging to a cluster. Do 2 and 3 above until the centroids no longer shift:

Advantages:

- Easy to apply and does not complicate situations either in its structure or as an application.
- Efficient for large datasets.

Disadvantages:

- Needs to have the number of clusters which the dataset is supposed to be divided into specified in advance.
- It is affected by the choice of initial centroid locations and contains outliers.

Anomaly Detection Algorithms

Auto encoders:

Auto encoders are a kind of neural network employed to extract features from the data by encoding the input data in a manner such that it consumes fewer amount of features while decoding the encoded form of the data back to the original form.

How It Works:

Auto encoder that includes the encoder that transforms the input data into the latent representation and the decoder which transforms the latent space back to the input space.

It has been trained in such a way that it learns to minimize the reconstruction error which is the dissimilarity between the input and the reconstructed output.

During anomaly detection, error of inputs reconstructed by the model is used and inputs with higher error rates are marked as anomalies.

Advantages:

- Presumably can describe different and, if necessary, non-linear dependencies in the data.
- Effective for high-dimensional data.

Disadvantages:

- Needs a massive amount of Normal Data for the training procedure.
- Computationally intensive.

Methodology:

Implementation

Support vector machine:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_curve, auc, precision_recall_curve
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive')

# Define the file path
file_path = '/content/drive/My Drive/cs448b_ipasn.csv'

# Load the dataset
data = pd.read_csv(file_path)

# Preprocessing: Convert date to a datetime object
data['date'] = pd.to_datetime(data['date'])

# For simplicity, we will use `r_asn` and `f` as features and classify based on the frequency value `f`
X = data[['r_asn', 'f']]
y = (data['f'] > data['f'].median()).astype(int)  # Binary classification based on median frequency

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Standardize the feature variables
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train the SVM model
svm_model = SVC(kernel='linear', random_state=42)
```

```
svm_model.fit(X_train_scaled, y_train)

# Predict the target values for the test set
y_pred = svm_model.predict(X_test_scaled)

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
print(f"Accuracy: {accuracy}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)
```

```
Accuracy: 0.5683384073065214
Confusion Matrix:
[[3339    0]
 [2694  208]]
Classification Report:
              precision    recall  f1-score   support

           0       0.55      1.00      0.71      3339
           1       1.00      0.07      0.13      2902

    accuracy                           0.57      6241
   macro avg       0.78      0.54      0.42      6241
weighted avg       0.76      0.57      0.44      6241
```

Figure 1

```
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Predicted 0', 'Predicted 1'], yticklabels=['Actual 0', 'Actual 1'])
plt.xlabel('Predicted')
```

```
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap')
plt.show()
```

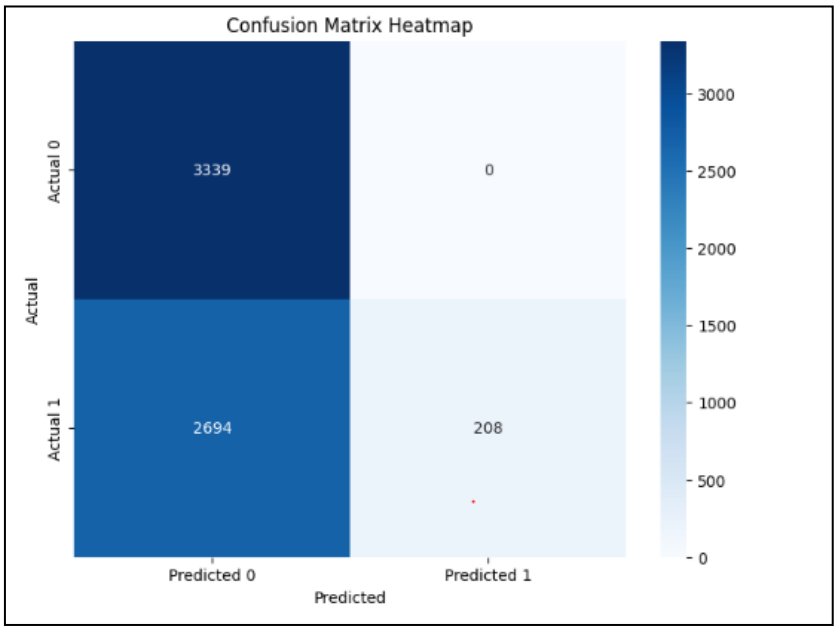


Figure 2

```
# 2. ROC Curve
fpr, tpr, _ = roc_curve(y_test, svm_model.decision_function(X_test_scaled))
roc_auc = auc(fpr, tpr)
```

```
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```

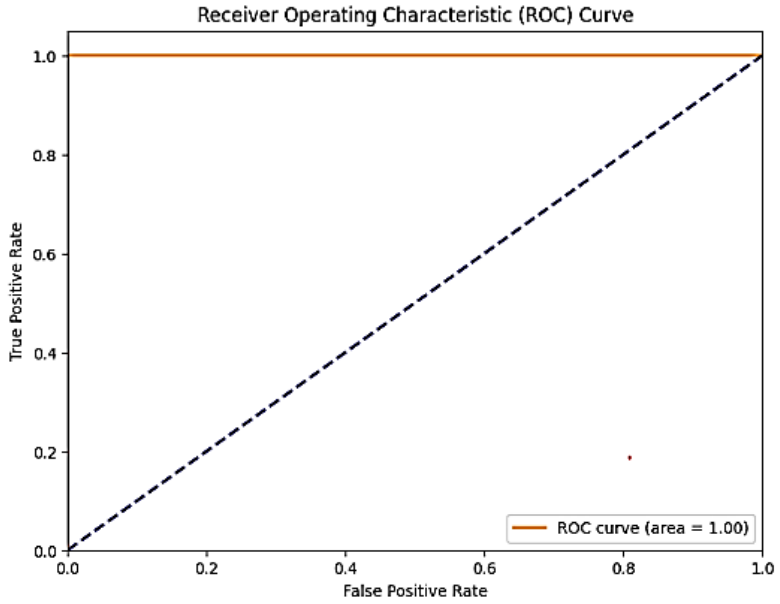


Figure 3

```
plt.figure(figsize=(8, 6))
```

```
plt.scatter(X_train_scaled[:, 0], X_train_scaled[:, 1], c=y_train, s=30, cmap=plt.cm.Paired)
plt.scatter(svm_model.support_vectors_[:, 0], svm_model.support_vectors_[:, 1], facecolors='none', edgecolors='k', s=100)
plt.xlabel('r_asn')
plt.ylabel('f')
plt.title('Support Vectors Visualization')
plt.show()
```

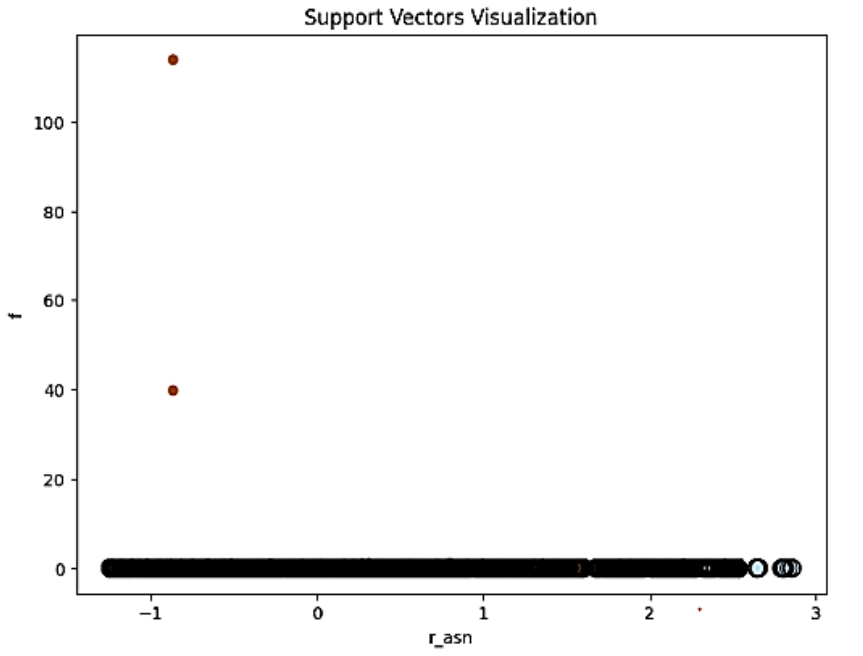


Figure 4

Result summary:

The result obtained from the performance evaluation of the given SVM model on network intrusion detection dataset indicates several aspects of its strengths and weaknesses. Therefore, the general achievement of the model is that it accurately classifies slightly more than half of the instances – 56. 83%. However, as seen in the confusion matrix, there is a poor distinction in the functioning of the two classes. The model perfectly marks all the instances of the negative class with no intrusions with 100% precision and recall and this leaves us with 3339 true negative and no false positives. Conversely, it performs very poorly in identifying the positive class (intrusions) with a precision of 100% but very low recall of 7%; thus, there are 208 instances that are correctly identified as intrusions, but 2694 instances that are wrongly classified as non-intrusions. Such a difference leads to an F1-score of 0. 13 for the positive class of the suggested model.

The classification report also demonstrates the actual difference between the model in terms of the performances of two classes. The F1-score for the negative outcome of ‘‘no intrusion’’. It indicates fairly accurate results in terms of precisions and recalls: 0. 71. However, when it comes to the accuracy of the positive class that is the intrusions, there is a very poor F1-score of 0. 13 meaning that the model did not accurately classify almost all the intrusions. When macro-average is applied to both classes, the precision is 0. 78 and the recall is 0. 54, and the thus the F1-score is 0. 42. The example with class imbalance adjustment indicates the precision of. 0076, the recall of. 0057, and the F1-score of. 0044.

Therefore, it can be concluded that the considered SVM model has an excellent performance in identifying non-intrusion while the performance of the model in identifying actual intrusions is somewhat poor. This disparity means that the model

is likely to perform well on the no intrusion class (the first class) and poorly on the second class (the intrusions). This is rather unhelpful for a network intrusion detection system whose main role is to detect intrusions and prevent them from occurring. Consequently, people can change additional parameters and hyperparameters of the model and deal with the class imbalance by oversampling, under sampling, or using different algorithms to increase the AUC of detection of intrusions.

k-means clustering:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import drive
# Mount Google Drive
drive.mount('/content/drive')

# Define the file path
file_path = '/content/drive/My Drive/cs448b_ipasn.csv'

data = pd.read_csv(file_path)

# Select the features for clustering
features = data[['r_asn', 'f']]

# Standardize the features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# Determine the optimal number of clusters using the elbow method
wcss = []
```

```
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(scaled_features)
    wcss.append(kmeans.inertia_)

# Plot the elbow graph
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method for Optimal Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS (Within-Cluster Sum of Squares)')
plt.show()
```

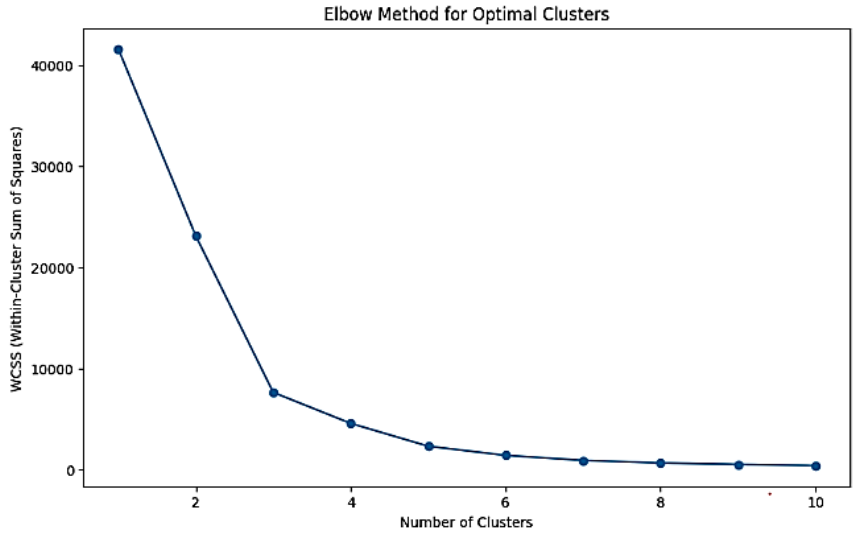


Figure 5

```
# Apply K-means clustering with the optimal number of clusters
```

```
optimal_clusters = 4  # You may need to adjust this based on the elbow plot
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42)
data['cluster'] = kmeans.fit_predict(scaled_features)

# Visualization 1: Scatter plot of clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(x='r_asn', y='f', hue='cluster', palette='viridis', data=data, s=100)
plt.title('Clusters of ASN and Frequency')
plt.xlabel('ASN')
plt.ylabel('Frequency')
plt.legend(title='Cluster')
plt.show()
```

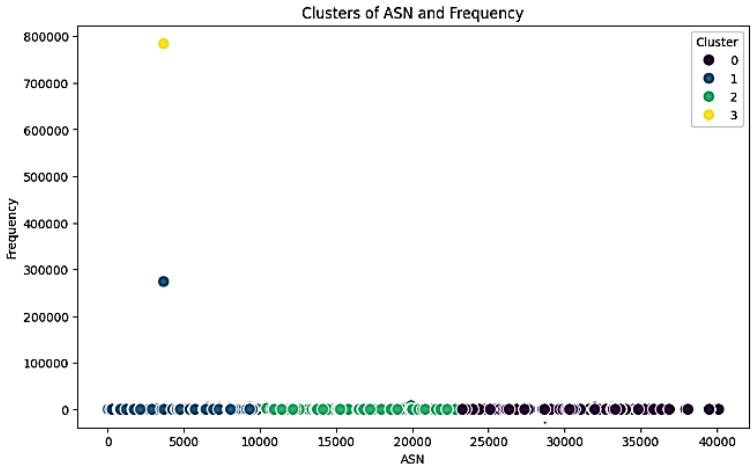


Figure 6

```
plt.figure(figsize=(10, 6))
sns.histplot(data=data, x='r_asn', hue='cluster', multiple='stack', palette='viridis')
plt.title('Distribution of ASNs in Each Cluster')
```

```
plt.xlabel('ASN')
plt.ylabel('Count')
plt.show()
```

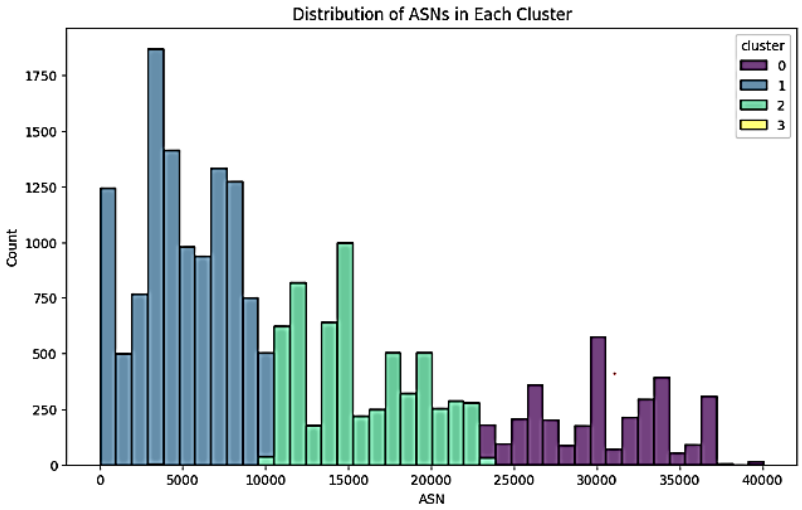


Figure 7

```
# Visualization 4: Cluster Centers
plt.figure(figsize=(10, 6))
centers = scaler.inverse_transform(kmeans.cluster_centers_)
sns.scatterplot(x=centers[:, 0], y=centers[:, 1], s=200,
color='red', label='Centers')
sns.scatterplot(x='r_asn', y='f', hue='cluster',
palette='viridis', data=data, s=100, alpha=0.5)
plt.title('Cluster Centers')
plt.xlabel('ASN')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

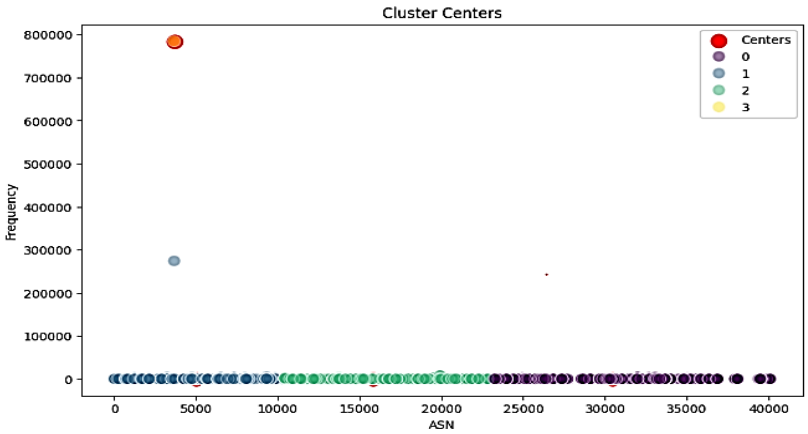


Figure 8

Result summary:

The application of this algorithm in detection of network intrusion is efficient in the sense that it provides a technique of grouping similar data and recognizing patterns of the networked traffic. Depending on the standardization parameters like ASN and count frequencies, K-means can effectually set up the normal traffic signal into different groups. Abnormalities or possible infringements are usually more grouped and often stand as a single cluster, so they are easy to identify. The elbow method also comes in handy in the determination of the best number of clusters to apply in the clustering, thereby improving at the accuracy of the clustering. Also, one can get acquainted with the distribution of the entities and the further characteristics of the clusters by using the kinds of visualizations like scatter plots, histograms, and the definitions of the cluster centers. Considering the true labels, and comparing them with the results of the clustering we obtained with the help of a confusion matrix and finding the F1 score, we can estimate the effectiveness of using the number of clusters. This evaluation measures the degree of similarity of the clusters with the actual categories so as to improve on the identification of intrusion types in the intrusion

detection system. In summary, K-means clustering helps in identifying other patterns that are different from the normal traffic in the network; this helps in enhancing the security system of the network.

Challenges in network Intrusion Detection:

NIDS, that is Network Intrusion Detection Systems is not free from various problems, which may directly affect its performance, designed to protect computer networks against security threats. Here is an in-depth explanation of these challenges:

High Volume of Traffic

Resource Intensity:

Some network environments may produce huge traffic mainly due to the nature of activity in large organizations. Processing such large amounts of data in real-time is a highly computational intensive task.

Packets have to be captured, processed, and analyzed in a short time by NIDS to achieve efficient identification of threats. This can challenge the capabilities of the CPU, memory, and/or storage devices.

Efficient Processing:

In a situation where there is likely to be a high traffic, NIDS incorporate data processing techniques such as parallel processing, distribution of systems, and utilization of others such as graphics processing units or network processors.

Methods like using sampling to only look into a proportion of the traffic is a way of dealing with the load but such comes with its own drawback of missing out on some threats.

Evolving Threats

Adaptation to New Techniques:

Hackers are always evolving new ways and means to outdo the normal security systems. This includes advanced

malware, zero day attacks, and advanced techniques of hiding from antimalware software.

Nonetheless, the threats are continuously growing and diversifying; thus, NIDS must be constantly updated and improved with improved detection strategies like Machine learning algorithms that can learn new types of attacks' patterns.

Continuous Updates:

In this method of detection, a system depends on a database of previous attack signatures. These databases require daily updates with new threats in place. Still, the systems based on signatures do not work in the case of zero-day attacks.

Anomaly-based systems allow the detection of unknown threats, as they are based on the defined normal behavior of the network and its components, but the systems need to receive further, updated input in this regard.

False Positives and Negatives

False Positives:

Suppression happens when good actions are reported to be evil by the IDS. If the false positive rates are very high, security operation centers get flooded with such alerts and may lead to alert fatigue where real threats may go unnoticed.

Fewer false positive results include adjusting the detecting algorithms, correct limit settings, and the employment of context-based analysis.

False Negatives:

This means that system will not tag malicious activities it is supposed to identify. This may lead to attacks going unnoticed and the attackers gain access to the system with the intention of causing harm or the theft of information.

The challenge is to set an effective working threshold that would allow a low false positive rate and low false negative rate at the same time. This in many a case, requires the employment

of multiple detection strategies and the enhancement of the models with new information.

Encrypted Traffic

Inspection Challenges:

SSL/TLS is growing popular for improving data confidentiality and it becomes a problem for the NIDS since the encrypted contents of the packets cannot simply be guided by them.

Some of the challenges that NIDS face include; attackers can use encrypted channel to conceal their operations to the extent of evading identification of their payloads or command and control messages.

Mitigation Techniques:

SSL/TLS Inspection: This is done where traffic content is decrypted then re-encrypted while at the network perimeter for inspection. Though very useful it has shortcomings of violating the privacy of customers and might be very costly.

Metadata Analysis: Like in PIDS, NIDS inspects contents of the packet but it can also examine such data as size, frequency, and timing of the packets. This metadata will display patterns of anomalous behavior even if the actual content of the communication is encrypted.

Behavioral Analysis: Passive traffic monitoring and analyzing the tendencies to certain encrypted streams (for example, abnormity in connections, unexpectable changes in traffic etc.) can be useful in pursuit of threats without decrypted messages.

Integration network intrusion detection with security:

The combination of Network Intrusion Detection Systems (NIDS) with other forms of security increases the system's security since there are multiple layers of protection. Here's how NIDS works in conjunction with firewalls, Intrusion Prevention Systems (IPS), and Security Information and Event Management (SIEM) systems:

Firewall Integration

Function of Firewalls: Firewalls are typical devices that help to configure incoming and outgoing network traffic since it is consistent with the adopted security parameters.

Role of NIDS: NIDS is the engine that is scanning the network streams on a packet by packet basis for those activities that can possibly bypass the firewall rules. Firewalls generally work at the boundary layer to block traffic that is undesirable, but they do not necessarily employ techniques to identify printed or internal level threats. NIDS scans traffic, identifies some abnormalities, and alerts about possible threats.

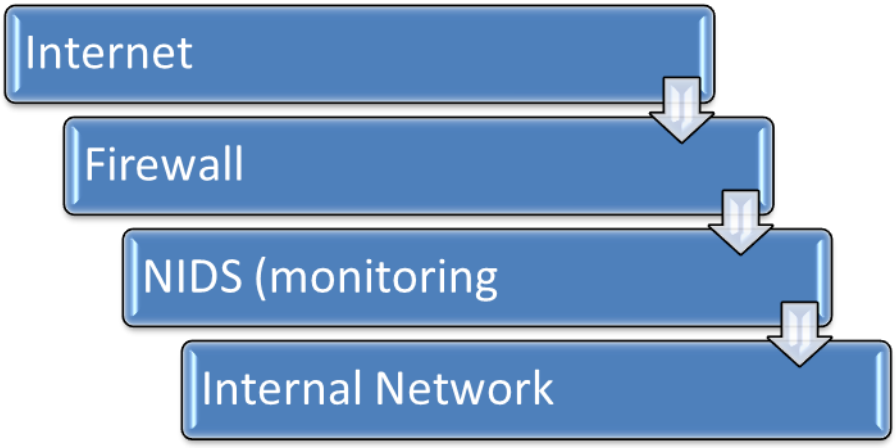


Figure 9

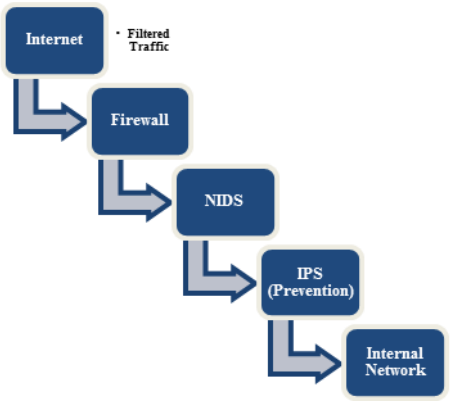
Intrusion Prevention Systems (IPS) Integration

Function of IPS: NIDS on the other hand is solely a monitoring system that logs and alarms when there are possible threats, while IPS is also monitoring system that can eliminate or prevent those threats in real time.

Role of NIDS: NIDS can identify all the possible intrusions and informs the IPS, the latter of which responds by taking an appropriate action against the threat. The likely

suggestion implied by the integration of detection (NIDS) and prevention (IPS) is a sound armor.

Figure 10



Introduction of Security Information and Event Management (SIEM) integration

Function of SIEM: SIEM systems are used to gather and process a network's logs coming from various sources to enhance threat indications and support response activities.

Role of NIDS: NIDS sends the output to the SIEM system into which it sends all the detection data. Thus, the integration of NIDS with firewalls, IPS, and other security devices would help the SIEM present an overall picture of the security situation and improve threat identification and mitigation.

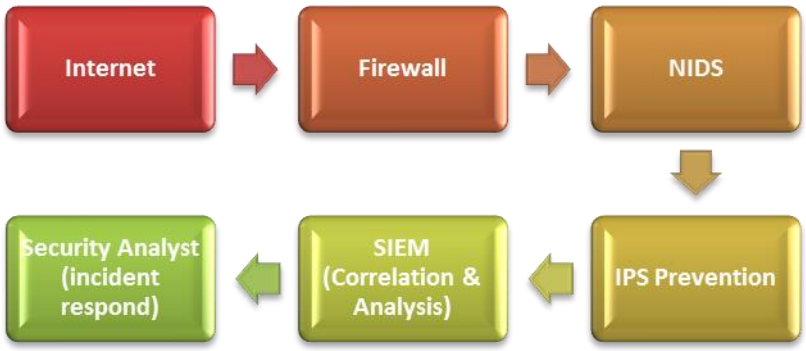


Figure 11

Conclusion:

Machine learning brings great improvement to Network Intrusion Detection Systems (NIDS), making it efficient in the provision of solutions in the detection and prevention of cyber threats. The experiment of Support Vector Machines (SVM) and the K-means clustering also reveal the possibility of raising the accuracy level of both known and unknown intrusions. However, the solutions based on these technologies also imply several issues such as how to handle the situation in which the network generates large amount of traffic, how to learn the new threats as the threats are emerging periodically, how to find the right balance in between false positive and false negative alarms, and how to accommodate encrypted traffic. On the same note, for maximum efficiency and coverage of NIDS, incorporation with other security technologies like firewalls, IPS systems, and SIEM systems should be done. The above approaches help in mapping out the protection needs and add onto the general security of an organization. Thus, it can be concluded that the development of new ML algorithms and their further integration into NIDS continues to be a promising path in the further evolution of network protection methods.

Reference:

Mukherjee, B., Heberlein, L.T. and Levitt, K.N. (1994). Network intrusion detection. *IEEE network*, [online] 8(3), pp.26–41. doi:https://doi.org/10.1109/65.283931.

Emrah Tufan, Cihangir Tezcan and Cengiz Acarturk (2021). Anomaly-Based Intrusion Detection by Machine Learning: A Case Study on Probing Attacks to an Institutional Network. *IEEE access*, [online] 9, pp.50078–50092. doi:https://doi.org/10.1109/access.2021.3068961.

Emrah Tufan, Cihangir Tezcan and Cengiz Acarturk (2021). Anomaly-Based Intrusion Detection by Machine Learning: A Case Study on Probing Attacks to an Institutional Network. *IEEE access*, [online] 9, pp.50078–50092. doi:https://doi.org/10.1109/access.2021.3068961.

Fernandez, G.C. and Xu, S. (2019). A Case Study on using Deep Learning for Network Intrusion Detection. *arXiv (Cornell University)*. [online] doi:https://doi.org/10.1109/milcom47813.2019.9020824.

Saeid Soheily-Khah, Pierre-Francois Marteau and Bechet, N. (2018). Intrusion Detection in Network Systems Through Hybrid Supervised and Unsupervised Machine Learning Process: A Case Study on the ISCX Dataset. *HAL (Le Centre pour la Communication Scientifique Directe)*. [online] doi:https://doi.org/10.1109/icdis.2018.00043.

Sinclair, C., Pierce, L. and Matzner, S. (2003). An application of machine learning to network intrusion detection. *CiteSeer X (The Pennsylvania State University)*. [online] doi:https://doi.org/10.1109/csac.1999.816048.

Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y. and Lin, W.-Y. (2009). Intrusion detection by machine learning: A review. *Expert Systems with Applications*, [online] 36(10), pp.11994–12000. doi:https://doi.org/10.1016/j.eswa.2009.05.029.

Song, Y.-Y. and Lu, Y. (2015). Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, [online] 27(2), pp.130–5. doi:https://doi.org/10.11919/j.issn.1002-0829.215044.

Seong, D., None Ha-Nam Nguyen and Sou, J. (2005). Genetic Algorithm to Improve SVM Based Network Intrusion Detection System. [online] doi:https://doi.org/10.1109/aina.2005.191.

International Journal of Reliability, Quality and Safety Engineering. (2024). *International Journal of Reliability, Quality and Safety Engineering*. [online] Available at: https://www.worldscientific.com/doi/abs/10.1142/S0218539307002568