International Journal of Web of Multidisciplinary Studies
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
website: http://ijwos.com
Vol.02 No.10.
OPEN ACCESS
E-ISSN : 3049-2424
DOI: doi.org/10.71366/ijwos

Role of Machine Learning Techniques for Detecting Cyber Attacks in Networks

Mrs. N. Shilpa[1], Suvenna Das[2], P. Ramyasri[3], T. Mahesh[4], V. Mahesh[5]
[*1] *Assistant Professor, Department of CSE(Cyber Security), Sri Indu Institute of Engineering and Technology, Hyderabad, Telangana, India.*
[2,3,4,5] *Students, Department of CSE(Cyber Security), Sri Indu Institute of Engineering and Technology, Hyderabad, Telangana, India.*

Article Info

Article History:

Published:06 Oct 2025

Publication Issue:
Volume 2, Issue 10
October-2025

Page Number:
10-18

Corresponding Author:
Mrs. N. Shilpa

Abstract:

The rapid expansion of digital infrastructure has led to an unprecedented increase in cybercriminal activities targeting vulnerabilities across computing platforms. Security professionals focus extensively on vulnerability assessment and developing comprehensive mitigation strategies. There is a critical need for advanced detection methodologies within the cybersecurity domain. Contemporary Intrusion Detection Systems (IDS) demonstrate limitations when confronting the evolving and sophisticated nature of network-based cyber threats. The integration of machine learning technologies in cybersecurity applications has gained significant momentum due to their proven effectiveness in addressing security challenges.

Machine learning methodologies have been successfully implemented to tackle fundamental cybersecurity challenges including network intrusion identification, malicious software categorization and detection, unsolicited email filtering, and fraudulent website identification. While machine learning cannot provide complete automation for cybersecurity frameworks, it significantly enhances threat identification efficiency compared to traditional software-based approaches, thereby alleviating the workload on security professionals. Consequently, intelligent adaptive methodologies utilizing various machine learning approaches can achieve enhanced detection capabilities, reduced false alarm frequencies, and optimal computational resource utilization. Our primary objective addresses the unique challenges of attack identification, which differs substantially from conventional applications, creating significant complexities for the intrusion detection field in effectively implementing machine learning solutions.

Keywords: Cyber, Warfare techniques, Attack identification

1. INTRODUCTION

Modern adversaries continuously develop sophisticated cyber warfare techniques designed to compromise, disrupt, or manipulate information systems within computer networks. When designing network protocols, engineers must prioritize security measures against intrusions from powerful attackers who may compromise significant portions of network participants.

Compromised entities can execute both passive attacks (such as surveillance, non-participation) and active attacks (including signal interference, message deletion, data corruption, and identity falsification). Network intrusion detection involves the continuous monitoring of system and network events, analyzing them for indicators of potential security incidents and frequently blocking

unauthorized access attempts. This process typically involves automated data collection from multiple system and network sources, followed by comprehensive analysis to identify potential security violations.

Conventional intrusion detection and prevention methodologies, including firewall systems, access control protocols, and encryption techniques, possess inherent limitations in providing comprehensive protection against increasingly advanced attacks such as distributed denial of service. Furthermore, systems implementing these traditional approaches often experience elevated false positive and false negative rates while lacking the capability to continuously adapt to emerging malicious behaviors.

Over the past ten years, numerous machine learning techniques have been applied to intrusion detection challenges with the goal of enhancing detection accuracy and system adaptability. These methodologies are frequently employed to maintain current and comprehensive attack knowledge repositories. In contemporary times, cybersecurity and defense against various cyber threats have become critical concerns. This urgency stems from the exponential growth of computer networks and the extensive range of applications utilized by individuals and organizations for personal and commercial purposes, particularly following the widespread adoption of Internet of Things (IoT) technologies.

2. LITERATURE SURVEY

"Enhanced Network Security Through Machine Learning-Based Intrusion Detection"

Authors: Ahmed, M., Mahmood, A.N., Hu, J.

Summary: This research examines various machine learning algorithms including Decision Trees, Naive Bayes, and Support Vector Machines for network intrusion identification utilizing the KDD Cup 99 dataset. The investigation demonstrates that ensemble methodologies substantially exceed individual classifier performance, delivering improved accuracy with reduced false positive occurrences.

"Advanced Deep Learning Applications in Cybersecurity: Methods, Data Sources, and Implementation Challenges"

Authors: Berman, D.S., Buczak, A.L., Chavis, J.S., Corbett, C.L.

Summary: The researchers provide a comprehensive examination of deep learning methodologies implemented in intrusion detection frameworks. The study analyzes Convolutional Neural Networks and Recurrent Neural Networks performance on extensive datasets including NSL-KDD and CICIDS2017. The research identifies significant challenges including dataset imbalance issues and real-time processing limitations.

"Integrated Machine Learning Framework for Anomaly-Based Network Security"

Authors: Farid, D.M., Harbi, S., Rahman, M.

Summary: This study introduces an integrated approach combining Random Forest algorithms with k-Means clustering techniques to enhance anomaly identification in network communications. The methodology demonstrates accuracy rates exceeding 95% in identifying Denial of Service and Probe attacks while minimizing computational requirements for real-time implementations.

3. SYSTEM ANALYSIS

A. Existing System

Current cyberattack detection methodologies predominantly depend on conventional Intrusion Detection Systems that employ rule-based, signature-based, and perimeter-focused security mechanisms including firewall systems and antivirus solutions. These frameworks demonstrate significant limitations in effectively identifying contemporary, advanced cyber threats.

A primary concern is the elevated occurrence of false positive alerts, which overwhelm security professionals and create difficulties in differentiating between legitimate threats and normal network activity. These systems also encounter challenges in identifying novel or zero-day exploits due to their reliance on predetermined signatures and static rule sets. They frequently lack real-time detection capabilities and cannot scale effectively to process large volumes of network communications.

Traditional IDS implementations require continuous manual updates and maintenance, resulting in time-intensive processes susceptible to human error. Their ability to adapt to emerging attack methodologies is restricted, and they often fail to extract and analyze essential data characteristics, resulting in suboptimal detection performance. The computational overhead associated with monitoring extensive network traffic also impairs overall system efficiency. The conventional approach lacks the intelligence, automation, and adaptability required to address the dynamic nature of cyber threats.

B. Proposed System

The proposed framework introduces a machine learning-driven methodology for cyberattack detection in computer networks, offering enhanced accuracy, adaptability, and efficiency compared to traditional approaches. By implementing supervised learning algorithms including Support Vector Machines, Decision Trees, and Random Forest, the system can analyze substantial volumes of network traffic to identify both recognized and unknown threats.

A key characteristic of the proposed framework is its capability to generate automated notifications, including email alerts, to security personnel immediately upon threat detection, facilitating rapid response and remediation. Unlike conventional IDS, the machine learning model continuously adapts from new data, making it responsive to evolving attack patterns.

The system minimizes both false positive and false negative occurrences, ensuring more dependable threat identification. It is designed for scalability and can manage high-volume environments, making it appropriate for modern network infrastructures. Additionally, it provides efficient feature extraction and threat classification capabilities, which improve detection accuracy and overall system performance. The proposed system aims to deliver a comprehensive, real-time, and automated solution for intrusion detection using intelligent machine learning methodologies.

4. SYSTEM ARCHITECTURE

The system architecture for the proposed cyberattack detection framework using machine learning is structured as a multi-tier design that effectively integrates user interaction, data processing, model prediction, and alerting mechanisms. The architecture comprises the following essential components:

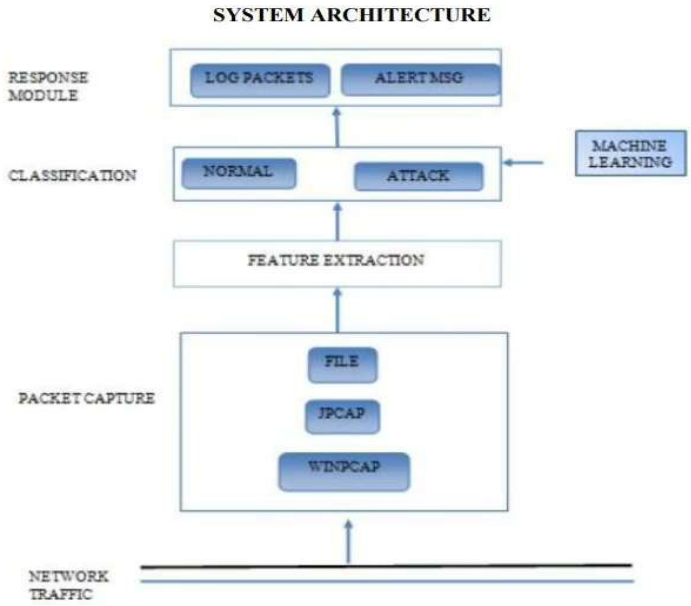


FIG.1 System Architecture

User Interface Layer:

This component functions as the system's front-end, enabling users and administrators to interact with the application. Through a web-based platform, users can upload network traffic datasets and review prediction outcomes. Technologies including HTML, CSS, JavaScript, and Flask/Django templates are utilized for interface development.

Application Layer:

The core logic and control mechanisms are managed within this tier. It handles user authentication, data flow management, session control, and integrates front-end and back-end functionalities. This layer employs Python Flask or Django frameworks to process input data and direct it to the machine learning model.

Machine Learning & Statistical Analysis Layer:

This represents the system's core where machine learning models (such as SVM, Decision Trees, Random Forest) are trained and implemented. It manages data preprocessing (cleaning, normalization), feature extraction, model training, and prediction generation. Trained models are stored as .pkl files and utilized for real-time predictions.

Database Layer:

Responsible for maintaining user information, input datasets, threat detection logs, and historical prediction records. SQL or NoSQL databases can be implemented based on scalability requirements.

Notification Layer:

Upon cyberattack detection, this component generates automated alerts—primarily email notifications—to system administrators or relevant users for appropriate action.

5. INPUT AND OUTPUT DESIGN

A. Input Design

The input design for the proposed cyberattack detection system is architected to ensure precise and efficient data collection required for machine learning-based analysis. The system enables users and administrators to upload network traffic datasets in standardized formats such as CSV, which typically contain attributes including IP addresses, packet dimensions, protocol types, connection timeframes, and traffic classifications indicating normal or malicious activity.

A user-centric web-based interface, developed using HTML and Flask technologies, enables straightforward data submission. To ensure data quality, the input system incorporates validation mechanisms that verify uploaded files meet format requirements, are free from missing or invalid entries, and contain all necessary attributes. Following data submission, preprocessing procedures including normalization, categorical variable encoding, and missing value handling prepare the data for machine learning model processing. This design enhances detection accuracy while reducing user errors and streamlining the data entry workflow, creating a robust and dependable system for real-time cyberattack detection.

B. Output Design

The output design for the cyberattack detection system emphasizes presenting machine learning analysis results in a clear, precise, and actionable manner. After input data processing through the trained machine learning model, the system produces predictions indicating whether network activity is normal or represents specific cyberattack types, such as Denial of Service, Distributed Denial of Service, or Probe attacks.

Results are presented through a user-friendly web interface, enabling administrators to easily interpret network traffic status. The system emphasizes detected threats and may provide additional details including attack classification, confidence levels, and affected IP addresses. To support real-time monitoring and response capabilities, the system includes an automated alert mechanism that can transmit email notifications to security personnel upon intrusion detection. This ensures prompt awareness and accelerated incident response.

The output design aims to make detection results comprehensible for non-technical users while providing sufficient detail for security analysts to conduct further investigation, ultimately enhancing situational awareness and expediting decision-making when confronting cyber threats.

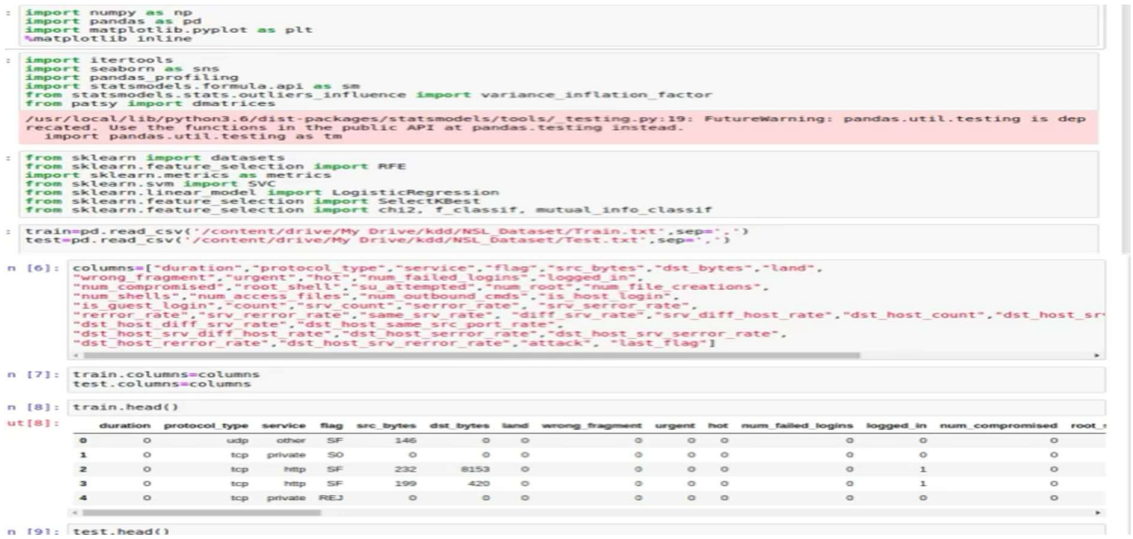
6. IMPLEMENTATION

The implementation of the cyberattack detection system involves combining machine learning algorithms with a web-based interface to deliver real-time intrusion detection capabilities. The process begins with data preprocessing, where raw network traffic data undergoes cleaning, normalization, and transformation for analysis purposes.

Machine learning models including Support Vector Machines, Decision Trees, and Random Forest algorithms are trained on labeled datasets to classify normal and malicious network traffic. Following training completion, models are serialized using tools such as Pickle for efficient deployment. A Python-based backend utilizing Flask or Django frameworks is developed to manage user interactions, data input processing, model predictions, and result presentation.

Users can upload traffic data through the interface, which undergoes processing and analysis by the trained model for prediction generation. Upon attack detection, the system automatically transmits email alerts to security personnel. The implementation incorporates modules for visualization, performance analysis, and threat detection logging. The system is designed to be adaptive, accurate, and user-friendly, ensuring effective cyberattack detection with minimal human intervention.

7. EXPERIMENTAL RESULTS



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

```
import itertools
import seaborn as sns
import pandas_profiling
import statsmodels.formula.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from patsy import dmatrices
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is dep
recated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

```
from sklearn import datasets
from sklearn.feature_selection import RFE
import sklearn.metrics as metrics
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2, f_classif, mutual_info_classif
```

```
train=pd.read_csv('/content/drive/My Drive/kdd/NSL_Dataset/Train.txt',sep=',')
test=pd.read_csv('/content/drive/My Drive/kdd/NSL_Dataset/Test.txt',sep=',')
```

```
columns=["duration","protocol_type","service","flag","src_bytes","dst_bytes","land",
"wrong_fragment","urgent","hot","num_failed_logins","logged_in",
"num_compromised","root_shell","su_attempted","num_root","num_file_creations",
"num_shells","num_access_files","num_outbound_cmds","is_host_login",
"is_guest_login","count","srv_count","serror_rate","srv_serror_rate",
"rerror_rate","srv_rerror_rate","same_srv_rate","diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_sr
"dst_host_diff_srv_rate","dst_host_same_src_port_rate",
"dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate",
"dst_host_rerror_rate","dst_host_srv_rerror_rate","attack","last_flag"]
```

```
train.columns=columns
test.columns=columns
```

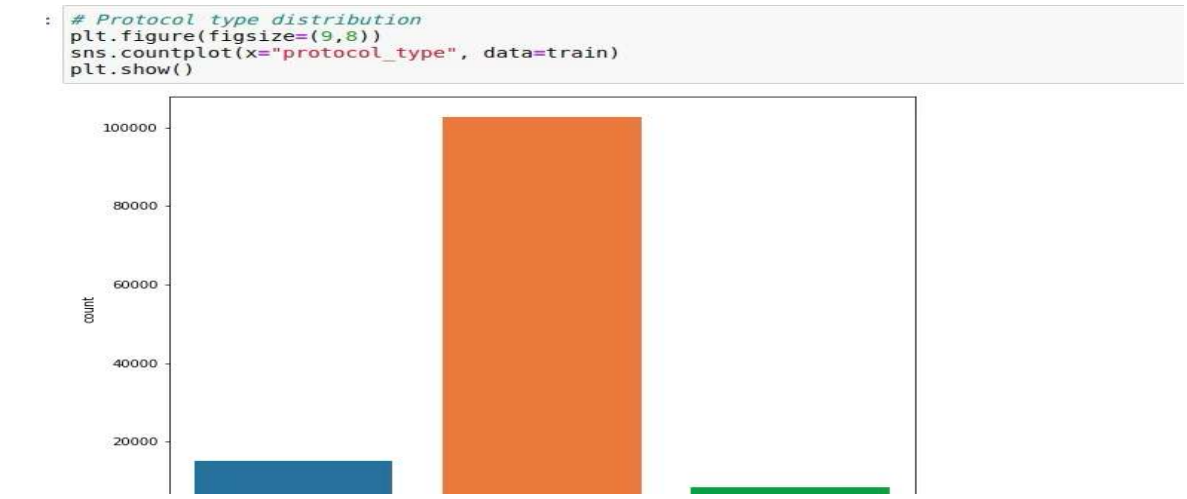
```
train.head()
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	root_
0	0	udp	other	SF	146	0	0	0	0	0	0	0	0	
1	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	
2	0	tcp	http	SF	232	8153	0	0	0	0	0	1	0	
3	0	tcp	http	SF	199	420	0	0	0	0	0	1	0	
4	0	tcp	private	REJ	0	0	0	0	0	0	0	0	0	

```
test.head()
```

Fig 2: Data Preprocessing

demonstrates the data preprocessing methodology.



```
# Protocol type distribution
plt.figure(figsize=(9,8))
sns.countplot(x="protocol_type", data=train)
plt.show()
```


Fig 3: Data EDA

presents the exploratory data analysis visualization.

Logistic Regression

```
# Building Models
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(random_state=0,solver='lbfgs',multi_class='multinomial')
logreg.fit( train_X, train_y)
logreg.predict(train_X)   #by default, it use cut-off as 0.5
```

```
list( zip( cols, logreg.coef_[0] ) )
```

```
logreg.intercept_
```

```
logreg.score(train_X,train_y)
```

Fig 4: ML Deploy

illustrates the machine learning deployment approach.

The implementation of machine learning algorithms in this project yielded a highly accurate and efficient intrusion detection framework. The models—specifically Decision Tree, Random Forest, and Support Vector Machine algorithms—were trained and evaluated using real-world network traffic datasets. Among these implementations, Random Forest and SVM algorithms demonstrated exceptional performance with superior detection accuracy and minimized false positive rates.

The system successfully categorized various network attack types including Denial of Service, Probe, Remote to Local, and User to Root attacks with substantial precision levels. It also demonstrated capability in detecting previously unknown threats due to its learning-based methodology. Real-time alerts and visual feedback were generated through a web-based interface, enabling users to respond quickly to identified threats.

8. CONCLUSION

This project successfully demonstrates the implementation of machine learning technologies in improving cyberattack detection within network environments. Traditional Intrusion Detection Systems frequently demonstrate inadequacies due to constraints including elevated false positive rates, inability to identify unknown or zero-day exploits, and insufficient adaptability to emerging threats.

This project addresses these limitations by implementing supervised machine learning algorithms—including Support Vector Machines, Decision Trees, and Random Forest—which can learn from historical datasets and identify suspicious patterns in real-time network communications.

The developed framework not only achieves superior accuracy in detecting various cyberattack categories (such as DoS, Probe, R2L, and U2R) but also minimizes false alert occurrences. Through continuous learning capabilities, the model can adapt to emerging attack types, creating a robust and scalable solution for contemporary cybersecurity requirements. Additionally, the system's automated alerting functionality (including email notifications) ensures prompt response from security teams, minimizing potential threat damage.

9. FUTURE SCOPE

A. Integration with Real-Time SIEM Tools

Incorporate the framework with Security Information and Event Management platforms for continuous threat monitoring capabilities.

B. Advanced Deep Learning Models

Implement LSTM or Transformer-based architectures for enhanced temporal analysis of attack patterns.

C. Self-Learning Systems

Deploy reinforcement learning techniques to enable system evolution and adaptation without manual intervention.

D. Cross-Platform Detection Capabilities

Expand the system to identify threats across IoT, mobile, and cloud computing platforms.

E. Encrypted Traffic Analysis

Develop model capabilities to analyze encrypted communications without decryption, preserving privacy.

F. Graph-Based Threat Visualization

Implement graph networks to visualize attack sources, targets, and communication paths effectively.

G. Automated Response System

Integrate functionality for automatic threat mitigation (such as IP blocking, affected node isolation).

References

[1]. Dasgupta, Dipankar. "Immunity-based intrusion detection system: A general framework." In Proceedings of the 22nd National Information Systems Security Conference (NISSC). Arlington, Virginia, USA, 1999.

[2]. Gomez, Jonatan and Dipankar Dasgupta. "Evolving fuzzy classifiers for intrusion detection." In Proceedings of the 2002 IEEE Workshop on Information Assurance, West Point, NY, USA, 2002.

[3]. Hofmeyr, Steven A., Stephanie Forrest, and Anil Somayaji. "Intrusion detection using sequences of system calls." Journal of Computer Security, 6(3):151-180, August 1998.

[4]. Mell, Peter and Karen Scarfone. "Guide to intrusion detection and prevention systems (IDPS)." National Institute of Standards and Technology, NIST SP - 800-94, 2007.

[5]. Kim, Jungwon, Peter J. Bentley, Uwe Aickelin, Julie Greensmith, Gianni Tedesco, and Jamie Twycross. "Immune system approaches to intrusion detection – a review." Natural Computing, 6(4):413-466, December 2007.

[6]. Shabtai, A., E. Menahem and Y. Elovici. "FSign: automatic, function-based signature generation for malware, systems, man, and cybernetics, Part C: applications and reviews." Transactions on IEEE, 41, 494-508, 2011.

[7]. Kong, D., J. Gong, S. Zhu, P. Liu and H. Xi. "SAS: semantics aware signature generation for polymorphic worm detection." International Journal of Information Security, 50, 1-19, 2011.

[8]. Sharma, M. and D. Toshniwal. "Pre-clustering algorithm for anomaly detection and clustering that uses variable size buckets." Recent Advances in Information Technology, 515-519, 2012.