Journal Homepage: www.ijarpr.com ISSN: 3049-0103 (Online)

International Journal of Advance Research Publication and Reviews

Vol 2, Issue 10, pp 139-146, October 2025

Network Intrusion Detection and Response System Using a Combination of ML and RL Techniques

Vijay D[1], *Vishnu PS*[2], *Sreeja K*[3]

[1] *Student Dept. of Computer Science & Engineering B.N.M Institute of Technology* Bangalore, Karnataka, India
vijaydragh@gmail.com
[2] *Student Dept. of Computer Science & Engineering B.N.M Institute of Technology* Bangalore, Karnataka, India
psvishnu888@gmail.com
[3] *Assistant Professor Dept. of Computer Science & Engineering B.N.M Institute of Technology* Bangalore, Karnataka, India
sreejaak@bnmit.in

ABSTRACT—

In the rapidly evolving landscape of cybersecurity, traditional network intrusion detection systems (NIDS) often struggle to keep pace with the complexity and volume of modern network attacks. This paper proposes a hybrid intrusion detection and response system that leverages the strengths of both machine learning (ML) which helps in Network Intrusion Detection and reinforcement learning (RL) techniques to enhance the adaptability of NIDS to respond to Network Intrusion. Specifically, a decision tree classifier is employed for initial intrusion detection, utilizing a widely recognized dataset that simulates a military network environment with 41 diverse features per connection. To complement the static nature of the ML model, reinforcement learning is integrated to dynamically adapt and optimize the system's response to detected intrusions, thereby reducing false positives and improving real-time reaction capabilities. The combination of these techniques enables both robust pattern recognition and continuous learning in dynamic network conditions. Experimental results demonstrate a high detection accuracy of 99.32% using the decision tree model, with RL further refining the system's ability to respond effectively to emerging threats. This hybrid approach offers a promising step toward building intelligent, self-learning cybersecurity systems capable of proactive defense in real-time environments.

Keywords— Network Intrusion Detection Systems (NIDS), decision trees, reinforcement learning, cybersecurity, network attacks, and machine learning

Introduction

In today's digital era, where networks form the backbone of almost every sector—ranging from defense to finance to healthcare.

The security of networked systems is of paramount importance. With the increasing number of connected devices and services, there has also been a significant rise in malicious activities targeting vulnerabilities in these systems. Cyber-attacks have become more sophisticated, frequent, and damaging, making it essential for organizations to adopt more intelligent and adaptive security mechanisms.

Traditional Network Intrusion Detection Systems (NIDS), which often rely on predefined rule sets or static signature-based detection, are becoming less effective in identifying new, previously unseen threats. These systems are prone to high false positive rates and lack the ability to adapt to evolving attack patterns. To overcome these limitations, modern NIDS solutions are being augmented with Machine Learning (ML) and Artificial Intelligence (AI) techniques, which have shown great promise in detecting complex attack patterns through data-driven models.

Among the various ML approaches, Decision Tree classifiers are particularly notable for their interpretability and effectiveness in classifying network connections based on labeled training data. They can efficiently differentiate between normal and malicious behavior using a set of extracted features. However, static ML models can still fall short in dynamic environments where attack strategies change over time.

To address this, Reinforcement Learning (RL) is introduced into the intrusion response component. Unlike supervised learning, RL allows the system to learn optimal response strategies through interaction with the environment. It can dynamically update its policies to mitigate threats in real-time, offering a more robust and intelligent security framework.
This paper presents a hybrid intrusion detection and response system that combines the classification power of Decision Trees with the adaptability of Reinforcement Learning in Responding to Intrusion Detection. The model is trained and evaluated using a benchmark dataset simulating a military network environment, where each TCP/IP connection is labeled as either normal or an attack, and described by 41 features (3 qualitative and 38

quantitative). Our experimental results demonstrate that this combined approach not only achieves high detection accuracy but also enhances the system's capability to respond proactively and intelligently to intrusions.

Literature Survey

Over the past two decades, Network Intrusion Detection Systems (NIDS) have evolved from simple rule-based filters to more advanced, intelligent systems powered by machine learning. The core idea behind NIDS is to detect unauthorized access or abnormal behavior in a network before significant damage occurs.

One of the foundational datasets for evaluating NIDS performance is the KDD Cup 1999 dataset, developed by MIT Lincoln Labs to simulate real-world military network scenarios with both normal and attack connections. It consists of 41 features and has been widely adopted in research for training and benchmarking NIDS models [1].

Machine learning (ML) techniques have shown promising results in enhancing NIDS capabilities. Among these, decision tree (DT) classifiers are notable for their interpretability and low computational overhead. They can efficiently classify connections as normal or malicious using labeled data. Farnaaz and Jabbar [2] demonstrated the effectiveness of random forest and decision tree models in achieving high classification accuracy for intrusion detection tasks. Similarly, Xie et al. [3] explored various decision tree variants for anomaly detection on KDD data and reported significant improvements over rule-based approaches.

Beyond traditional ML, the integration of reinforcement learning (RL) into NIDS has gained attention in recent years. Sutton and Barto [4] provide a comprehensive foundation for RL, detailing how agents can learn optimal policies through reward feedback. RL has been used to enhance the adaptability of NIDS by enabling systems to autonomously choose optimal responses to detected threats in real time.

Kim et al. [5] introduced a hybrid model combining anomaly-based detection with misuse detection using reinforcement learning for policy optimization. The system dynamically learned which responses were most effective against different types of attacks, reducing false positives and improving response efficiency.

Garcia-Teodoro et al. [6] reviewed various anomaly-based detection techniques and highlighted the importance of adapting detection and response mechanisms based on network behavior patterns. They emphasized that combining statistical models with intelligent learning systems could significantly improve NIDS performance.
In another study, Shone et al. [7] proposed a deep learning approach for NIDS, using autoencoders to learn representations of network traffic, and a classifier to detect intrusions. While deep learning offers superior accuracy, it often lacks interpretability and requires substantial computational resources, making decision trees a more lightweight and practical solution for real-time systems.

In terms of response strategies, research by Khan et al. [8] showed that static rule-based responses often lag in dynamic environments. RL models like **Deep Q Networks** allow the system to update its response strategies based on real-time feedback from the environment, making the NIDS proactive rather than reactive.

Moreover, Tsai et al. [9] conducted a comprehensive review comparing different ML approaches for NIDS and concluded that hybrid models—those combining detection and response—offer the best trade-off between accuracy, efficiency, and scalability. These findings collectively indicate that a hybrid approach combining ML (for accurate detection) and RL (for dynamic response) is both practical and effective. This motivated the development of the system proposed in this paper, which leverages the strengths of both to build an adaptive and intelligent intrusion detection and response framework.

Proposed Solution

To address the limitations of traditional intrusion detection systems, this paper proposes a hybrid Intrusion Detection and Response System (IDRS) that leverages the power of both Machine Learning (ML) and Reinforcement Learning (RL). The core idea is to design a system that not only detects intrusions accurately but also learns to respond adaptively based on past interactions with threats in real-time network environments Using Reinforcement Learning.

The architecture of the proposed system consists of two major components: a Detection Module using a Decision Tree classifier and a Response Module powered by Reinforcement Learning. These components work together to create a closed-loop security framework that continuously monitors, detects, and responds to suspicious activities. The system is trained and tested on a well-known benchmark dataset that simulates a military network environment, namely the KDD Cup 1999 dataset. This dataset includes various types of network intrusions and normal traffic, with each TCP/IP connection represented by 41 features—3 categorical and 38 numerical. Each record is labeled as either "normal" or an "attack" with a specific type.

For the detection phase, a Decision Tree classifier is employed due to its simplicity, interpretability, and high classification performance. The model is trained on preprocessed data where categorical variables are encoded and numerical variables normalized.
The Decision Tree learns to distinguish between normal and malicious connections by generating a set of decision rules based on the input features. Once trained, it can classify new, unseen connections with a high degree of accuracy.

However, detecting an intrusion is only part of the solution. To effectively handle threats, the system includes a Reinforcement Learning-based response mechanism that determines the most suitable action to take after an intrusion is detected. This RL agent observes the system's state—which includes parameters like attack type, severity, past responses, and system load—and selects from a set of possible actions such as blocking the IP address.

The agent is trained using Deep-Q-Networks, where it learns a policy through exploration and reward feedback. Rewards are assigned based on the effectiveness of the response—for example, successfully preventing further malicious activity yields positive rewards, while incorrect or delayed responses incur penalties. Over time, the agent becomes more proficient at selecting optimal responses to different types of intrusions.

The interaction between the ML and RL modules ensures that the system not only detects threats with high accuracy (99.32%) but also responds intelligently based on contextual understanding and previous experiences. This makes the system adaptable to new and evolving threats, reducing false positives and improving the overall robustness of network security[9].

In summary, the proposed hybrid solution enhances traditional NIDS by incorporating learning-driven decision-making at both the detection and response stages. It mimics human expert behavior in analyzing threats and formulating actions, thereby bridging the gap between automation and intelligent cybersecurity defense.

System Desing And Development

The proposed Intrusion Detection and Response System (IDRS) has been designed to combine the strengths of both Machine Learning (ML) and Reinforcement Learning (RL), enabling intelligent and adaptive defense against network intrusions. The system is modular and consists of three primary stages: data preprocessing, intrusion detection, and dynamic response generation. Its foundation is built on a well-known benchmark dataset that simulates a U.S. Air Force LAN under both normal and attack conditions.

The dataset contains 41 features for each TCP/IP connection, including three categorical and 38 numerical attributes. These features reflect various aspects of network traffic, such as connection duration, the number of bytes transferred, login attempts, and service types. Before training, categorical features like protocol_type, service, and flag are encoded using one-hot encoding, while all numerical features are normalized to a consistent scale to ensure uniform input for both the Decision Tree and RL models.

Following best practices from Tsai et al. [9], a correlation-based feature selection process is applied to reduce dimensionality and retain only the most informative attributes. The dataset is then split into training (80%), and testing (20%) sets for model evaluation.

The detection module utilizes a Decision Tree classifier, implemented using the Scikit-learn library [2], chosen for its high interpretability, low computational cost, and excellent performance in network traffic classification. Decision Trees have consistently demonstrated their ability to effectively detect intrusions with fast inference speeds and minimal preprocessing, making them suitable for real-time NIDRS applications[9]. In our experiments, the Decision Tree model achieved an accuracy of 99.32%, corroborating findings by Farnaaz and Jabbar [2], who showed similar performance levels using tree-based models on KDD99 data. The model learns a set of rule-based paths that distinguish between normal and various types of malicious traffic.

To complement the static nature of the detection system, a dynamic, learning-based response mechanism is introduced using Deep Q Networks Reinforcement Learning Algorithm. The RL agent is designed to evaluate the current system.

A custom reward function is designed to promote timely and effective responses while discouraging false positives and unnecessary interventions. For example, a correct block action for a severe threat is highly rewarded, while ignoring a harmful connection result in a penalty. Over hundreds of training episodes, the RL agent improves its policy using the Bellman equation and converges toward optimal response strategies.

Here Incoming network connections are classified in real time by the Decision Tree module. If an intrusion is detected, the RL agent takes over to determine and execute an appropriate response. The system logs the outcome of each action to continuously update the agent's learning process. The System Design for the proposed system is shown in Fig. 1.

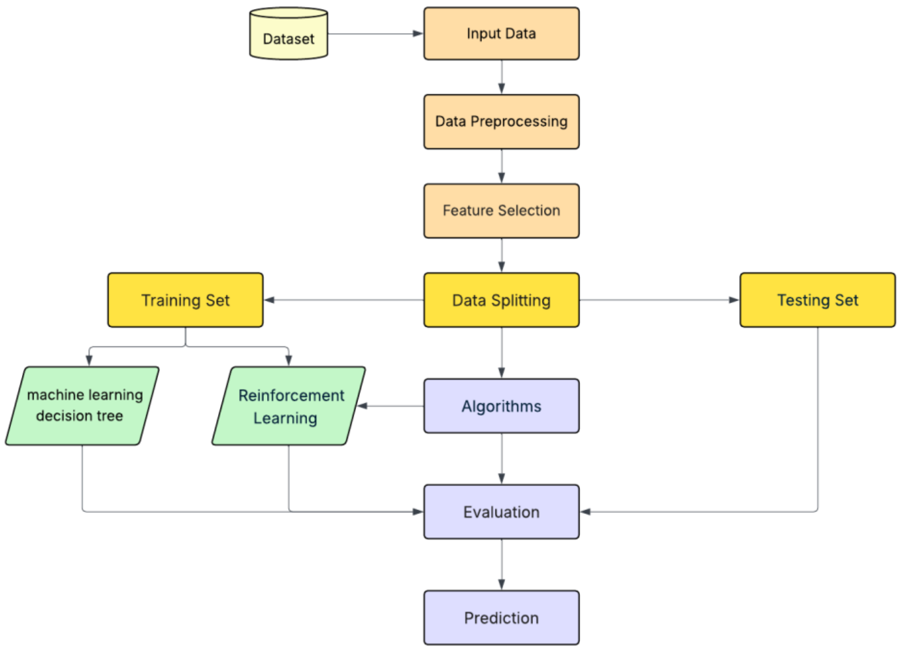


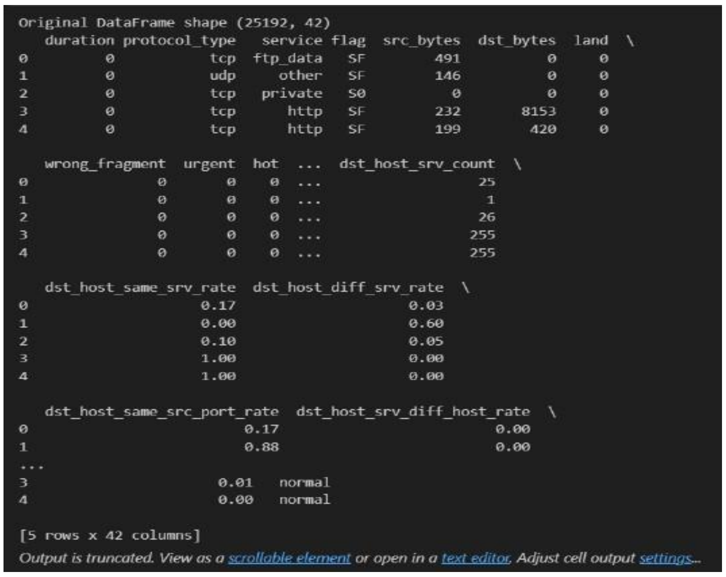
Fig. 1. System Design of NIDRS

This design aligns with the hybrid frameworks, where adaptive, learning-based systems outperform traditional NIDRS in dynamic and evolving attack scenarios. This hybrid approach ensures a more proactive and autonomous defense strategy, reducing the need for human intervention and strengthening cybersecurity infrastructure[9].

Results and Discussion

The Results and Discussion section presents the evaluation of the proposed Network Intrusion Detection and Response System (NIDRS), which integrates Deep Q-Network (DQN) reinforcement learning with real-time packet analysis. The model was assessed using both simulated and live network traffic. Key performance metrics: detection accuracy, response efficiency, and decision-making effectiveness—are analyzed. Comparative results with traditional machine learning models, along with the benefits of reinforcement learning for adaptive intrusion response, are discussed here.

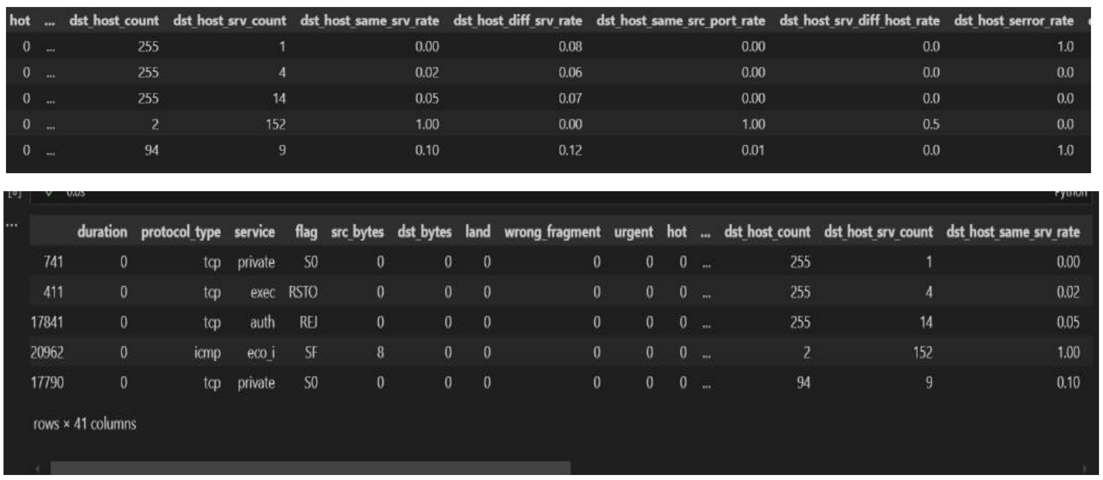
In Fig. 2. We can see the overview of the dataset used before preprocessing. The dataset contains 41 features and one target feature, ‘class’.



```
Original DataFrame shape (25192, 42)
   duration protocol_type   service flag  src_bytes  dst_bytes  land  \
0         0           tcp  ftp_data   SF        491          0     0
1         0           udp     other   SF        146          0     0
2         0           tcp   private   S0          0          0     0
3         0           tcp      http   SF        232       8153     0
4         0           tcp      http   SF        199        420     0

   wrong_fragment  urgent  hot  ...  dst_host_srv_count  \
0               0       0    0  ...                  25
1               0       0    0  ...                   1
2               0       0    0  ...                  26
3               0       0    0  ...                 255
4               0       0    0  ...                 255

   dst_host_same_srv_rate  dst_host_diff_srv_rate  \
0                    0.17                    0.03
1                    0.00                    0.60
2                    0.10                    0.05
3                    1.00                    0.00
4                    1.00                    0.00

   dst_host_same_src_port_rate  dst_host_srv_diff_host_rate  \
0                         0.17                         0.00
1                         0.88                         0.00
...
3                      0.01   normal
4                      0.00   normal

[5 rows x 42 columns]
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Fig. 2. Overview of the Dataset Initially

In Fig. 3. And Fig.. we can see the results of data Preprocessing of the initial dataset. Data Preprocessing is essential to train the model efficiently.



```
hot ... dst_host_count dst_host_srv_count dst_host_same_srv_rate dst_host_diff_srv_rate dst_host_same_src_port_rate dst_host_srv_diff_host_rate dst_host_serror_rate
  0 ...            255                  1                   0.00                   0.08                        0.00                         0.0                  1.0
  0 ...            255                  4                   0.02                   0.06                        0.00                         0.0                  0.0
  0 ...            255                 14                   0.05                   0.07                        0.00                         0.0                  0.0
  0 ...              2                152                   1.00                   0.00                        1.00                         0.5                  0.0
  0 ...             94                  9                   0.10                   0.12                        0.01                         0.0                  1.0
```

```
       duration protocol_type service  flag src_bytes dst_bytes land wrong_fragment urgent hot ... dst_host_count dst_host_srv_count dst_host_same_srv_rate
  741         0           tcp private    S0         0         0    0              0      0   0 ...            255                  1                   0.00
  411         0           tcp    exec  RSTO         0         0    0              0      0   0 ...            255                  4                   0.02
17841         0           tcp    auth   REJ         0         0    0              0      0   0 ...            255                 14                   0.05
20962         0          icmp   eco_i    SF         8         0    0              0      0   0 ...              2                152                   1.00
17790         0           tcp private    S0         0         0    0              0      0   0 ...             94                  9                   0.10

rows × 41 columns
```

Fig. 3. Data Preprocessing on Initial Dataset

We can see in Fig. 4. The contribution of different features in determining the values in target column. We can observe that only certain features such as 'src_bytes', 'dst_bytes', 'protocol_type', 'duration','service','flag' among the 41 available features in the initial dataset determine the value in target column.

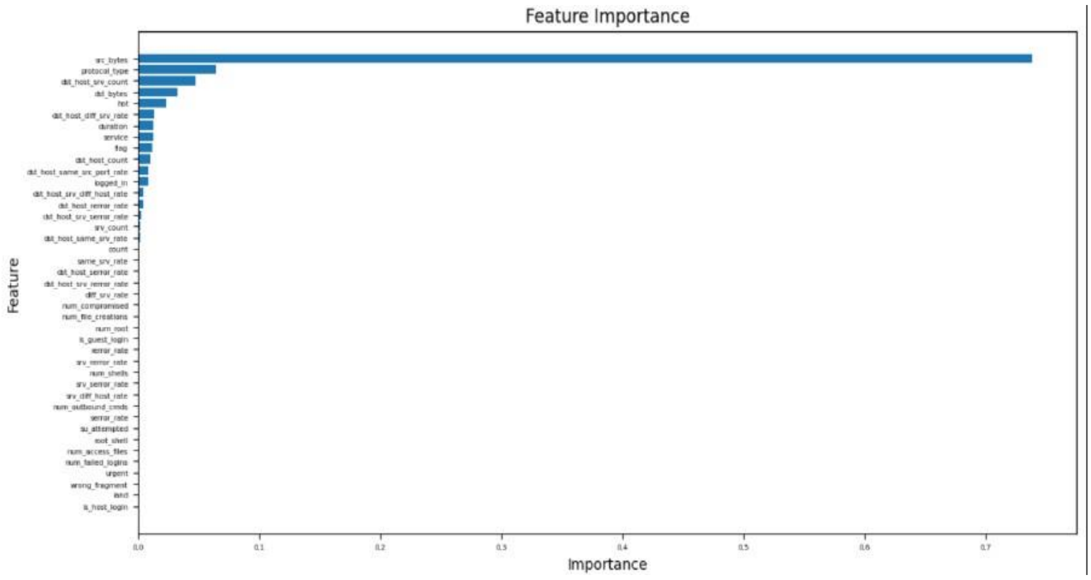
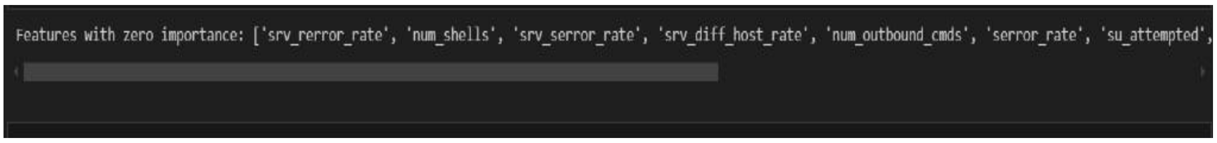


Fig. 4. Determining Feature Importance

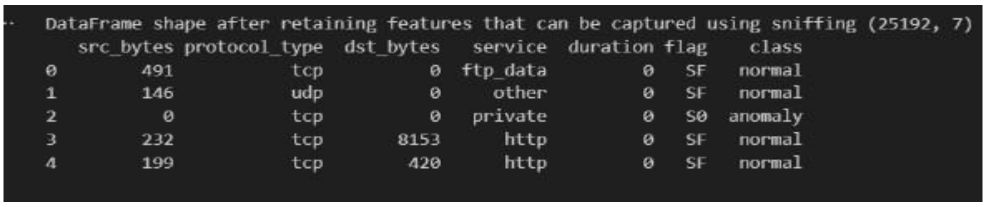
In Fig. 5. Shows the list of features with zero importance which can be omitted from consideration using dimensionality reduction for further analysis.



```
Features with zero importance: ['srv_rerror_rate', 'num_shells', 'srv_serror_rate', 'srv_diff_host_rate', 'num_outbound_cmds', 'serror_rate', 'su_attempted',
```

Fig. 5. Features with Zero Importance

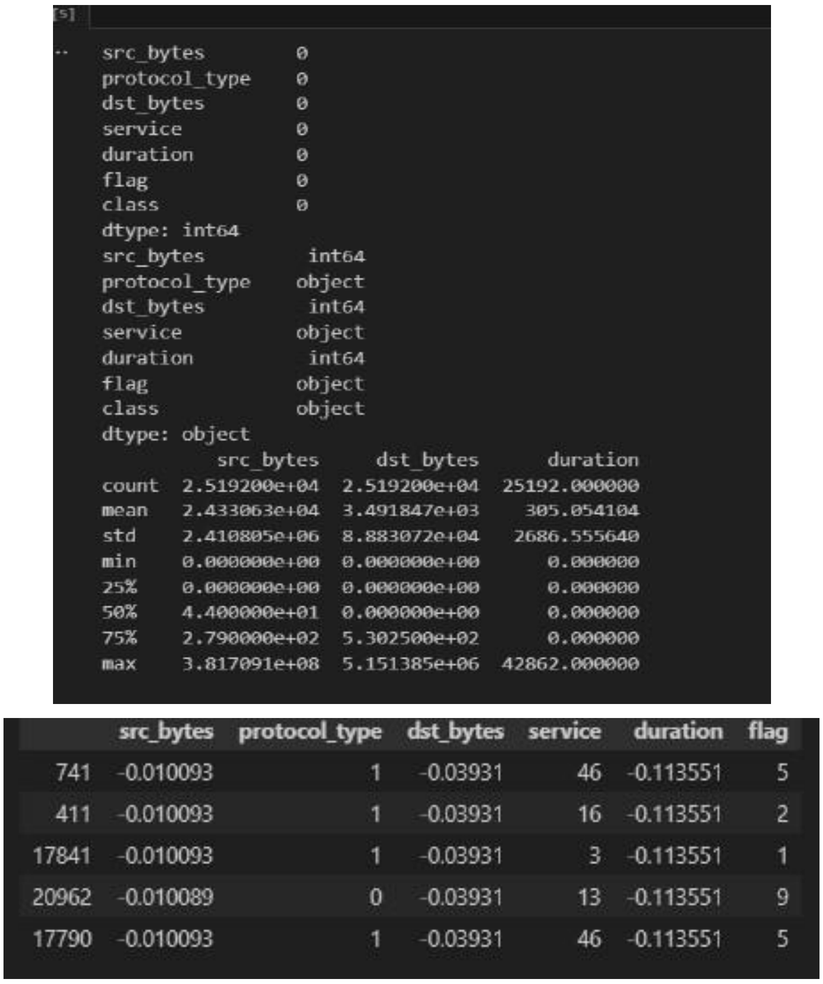
We have obtained a modified dataset involving the features that were not the part of list containing 'features with zero importance' and also can be analyzed by sniffing the network packets in real time. This is shown in Fig. 6.



```
DataFrame shape after retaining features that can be captured using sniffing (25192, 7)
   src_bytes protocol_type  dst_bytes   service  duration flag    class
0        491           tcp          0  ftp_data         0   SF   normal
1        146           udp          0     other         0   SF   normal
2          0           tcp          0   private         0   S0  anomaly
3        232           tcp       8153      http         0   SF   normal
4        199           tcp        420      http         0   SF   normal
```

Fig. 6. Overview of modified dataset Obtained

Fig. 7. Shows the preprocessing of the dataset obtained by retaining the features that can be obtained by sniffing the network packets and also played a important role in determining the values in target column ‘class’.



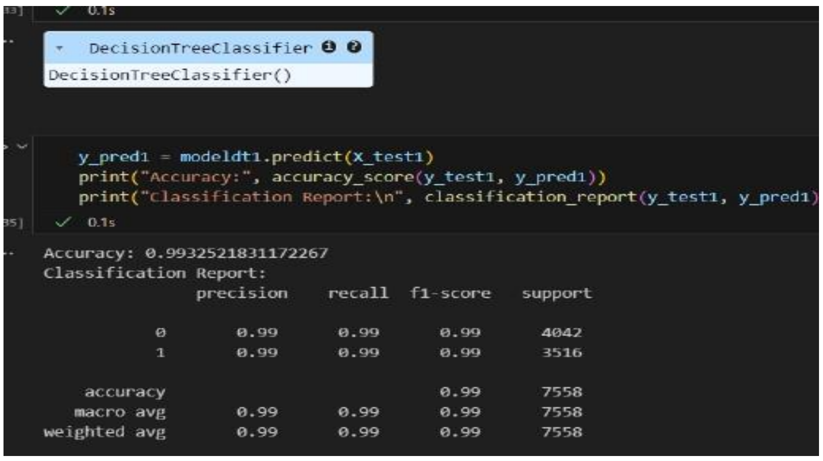
```
src_bytes        0
protocol_type    0
dst_bytes        0
service          0
duration         0
flag             0
class            0
dtype: int64
src_bytes         int64
protocol_type    object
dst_bytes         int64
service          object
duration          int64
flag             object
class            object
dtype: object
          src_bytes     dst_bytes      duration
count  2.519200e+04  2.519200e+04  25192.000000
mean   2.433063e+04  3.491847e+03    305.054104
std    2.410805e+06  8.883072e+04   2686.555640
min    0.000000e+00  0.000000e+00      0.000000
25%    0.000000e+00  0.000000e+00      0.000000
50%    4.400000e+01  0.000000e+00      0.000000
75%    2.790000e+02  5.302500e+02      0.000000
max    3.817091e+08  5.151385e+06  42862.000000
```

	src_bytes	protocol_type	dst_bytes	service	duration	flag
741	-0.010093	1	-0.03931	46	-0.113551	5
411	-0.010093	1	-0.03931	16	-0.113551	2
17841	-0.010093	1	-0.03931	3	-0.113551	1
20962	-0.010089	0	-0.03931	13	-0.113551	9
17790	-0.010093	1	-0.03931	46	-0.113551	5

Fig. 7. Preprocessing of the Modified dataset

Fig. 8. Shows the usage of Traditional ML Model Decision Tree Classifier in detecting the Network Intrusion by analyzing the features associated with respect to each record and classifying the output as ‘anomaly’ or ‘normal’. Since Decision Tree Classifier produced higher accuracy compared to other Traditional ML Models it was utilized in detecting the network intrusion. The Decision Tree Classifier is proving the accuracy upto 99% which can be seen in Classification Report Present in Fig. 8.

The value predicted by this model is further passed to Deep Q Network Reinforcement Model that provides response from the system after analyzing the flow of network packets.



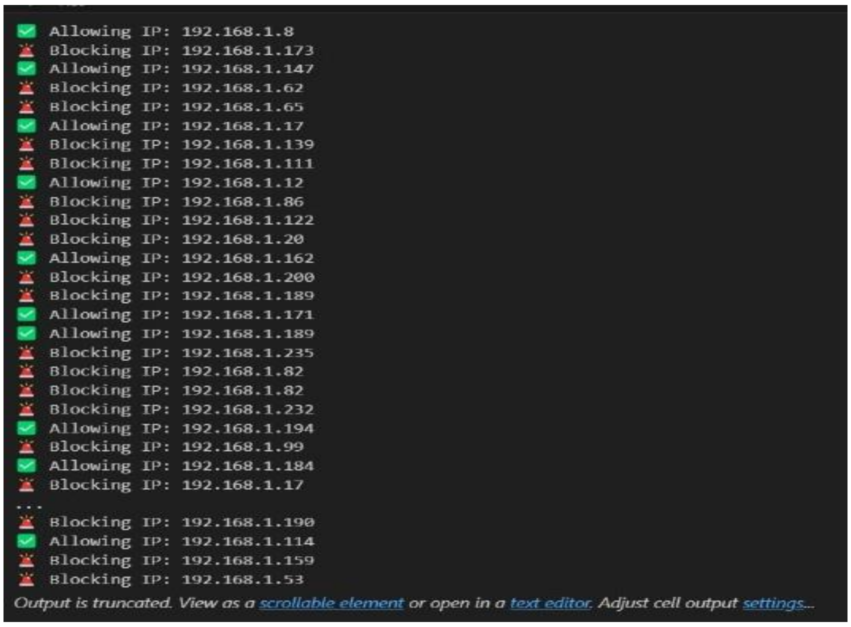
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
y_pred1 = modeldt1.predict(x_test1)
print("Accuracy:", accuracy_score(y_test1, y_pred1))
print("Classification Report:\n", classification_report(y_test1, y_pred1)
```

```
Accuracy: 0.9932521831172267
Classification Report:
               precision    recall  f1-score   support

           0       0.99      0.99      0.99      4042
           1       0.99      0.99      0.99      3516

    accuracy                           0.99      7558
   macro avg       0.99      0.99      0.99      7558
weighted avg       0.99      0.99      0.99      7558
```

Fig. 8. Usage of DecisionTreeClassifier ML Model

In Fig. 9. We can the Output of Deep Q Network Reinforcement model which provides response the flow of network packets by either Blocking if the network packet is found to be anomalous or allows through the network is it is found to be ‘normal’.



```
✅ Allowing IP: 192.168.1.8
🚨 Blocking IP: 192.168.1.173
✅ Allowing IP: 192.168.1.147
🚨 Blocking IP: 192.168.1.62
🚨 Blocking IP: 192.168.1.65
✅ Allowing IP: 192.168.1.17
🚨 Blocking IP: 192.168.1.139
🚨 Blocking IP: 192.168.1.111
✅ Allowing IP: 192.168.1.12
🚨 Blocking IP: 192.168.1.86
🚨 Blocking IP: 192.168.1.122
🚨 Blocking IP: 192.168.1.20
✅ Allowing IP: 192.168.1.162
🚨 Blocking IP: 192.168.1.200
🚨 Blocking IP: 192.168.1.189
✅ Allowing IP: 192.168.1.171
✅ Allowing IP: 192.168.1.189
🚨 Blocking IP: 192.168.1.235
🚨 Blocking IP: 192.168.1.82
🚨 Blocking IP: 192.168.1.82
🚨 Blocking IP: 192.168.1.232
✅ Allowing IP: 192.168.1.194
🚨 Blocking IP: 192.168.1.99
✅ Allowing IP: 192.168.1.184
🚨 Blocking IP: 192.168.1.17
...
🚨 Blocking IP: 192.168.1.190
✅ Allowing IP: 192.168.1.114
🚨 Blocking IP: 192.168.1.159
🚨 Blocking IP: 192.168.1.53
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Fig. 9. Testing of the Model on Real time Network Traffic

Thus we can see a system where Traditional ML Model Decision Tree Classifier is used to detect the Network Intrusion by analyzing the network packets in real time by sniffing the network packets and the output from the traditional ML Model is used in training the Deep Q Network to respond to the network traffic in real time and provide appropriate response. Thus a Hybrid Model of Traditional ML Models as well as Reinforcent Learning was utilized in developing a system where intrusion detection in integrated with respose system in Hybrid architecture.

Conclusion And Future Enhancements

In this paper, we propose a hybrid Network Intrusion Detection and Response System (NIDRS) that integrates machine learning and reinforcement learning techniques to create an intelligent, adaptive, and automated cybersecurity framework. The detection module, powered by a decision tree classifier, efficiently identifies abnormal network traffic and classifies it into specific types of intrusions based on a widely accepted military network simulation dataset. With an achieved detection accuracy of 99.32%, the model demonstrates strong potential for real-time intrusion classification with minimal false positives.

To complement detection, the system incorporates a Deep Q Networks based reinforcement learning agent to dynamically determine the most appropriate response to detected intrusions. This approach allows the system to go beyond static, rule-based responses and instead adapt its behavior based on the threat context and past response effectiveness. The integration of these modules establishes a closed-loop system capable of continuous learning, self-improvement, and intelligent threat mitigation.

The hybrid architecture addresses the limitations of traditional intrusion detection systems by enabling not just accurate threat identification but also timely and context-aware responses. It provides a foundation for building autonomous and self-healing cybersecurity systems capable of evolving alongside ever-changing attack patterns. The synergy of machine learning and reinforcement learning offers a flexible and robust mechanism to minimize response latency, reduce manual intervention, and enhance the overall network security posture.

Looking ahead, several opportunities exist to further enhance the system’s capabilities. Integrating real-time threat intelligence feeds and behavioral profiling can enable better detection of zero-day attacks and insider threats. The inclusion of natural language processing (NLP) could also assist in automated threat report generation and alert correlation across multiple systems.

Moreover, deploying the system in cloud-native or edge computing environments will enable scalable and decentralized protection, especially beneficial for large-scale IoT and distributed networks. Continuous feedback from real-world deployment can be used to retrain the models and update the RL policy dynamically, ensuring long-term adaptability.

In conclusion, the proposed ML-RL-based hybrid IDRS sets a promising direction for the future of intelligent cybersecurity solutions. By combining the speed and accuracy of supervised learning with the adaptability and decision-making strength of reinforcement learning, the system represents a critical step toward proactive, autonomous, and resilient network defense mechanisms.

REFERENCES

1. Stolfo, S., et al. (1999). "KDD Cup 1999 Dataset." UCI KDD Archive.
2. Farnaaz, N., & Jabbar, M. A. (2016). "Random forest modeling for network intrusion detection system." *Procedia Computer Science*, Elsevier.
3. OpenAI. (2024). "OpenAI Gym Toolkit Documentation." https://www.gymlibrary.dev
4. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*, MIT Press.
5. Kim, G., Lee, S., & Kim, S. (2014). "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection." *Expert Systems with Applications*, Elsevier.
6. Garcia-Teodoro, P., et al. (2009). "Anomaly-based network intrusion detection: Techniques, systems and challenges." *Computers & Security*, Elsevier.
7. Shone, N., et al. (2018). "A deep learning approach to network intrusion detection." *Future Generation Computer Systems*.
8. Khan, L., et al. (2007). "A new intrusion detection system using support vector machines and hierarchical clustering." *The VLDB Journal*, Springer.
9. Tsai, C. F., et al. (2009). "Intrusion detection by machine learning: A review." *Expert Systems with Applications*.
10. Heady, R., Luger, G., Maccabe, A., & Servilla, M. (1990). "The architecture of a network level intrusion detection system." Los Alamos National Laboratory, Technical Report LA-SUB-93-219.
11. Liu, H., & Lang, B. (2019). "Machine learning and deep learning methods for intrusion detection systems: A survey." Applied Sciences, 9(20), 4396.
12. Ahmed, M., Mahmood, A. N., & Hu, J. (2016). "A survey of network anomaly detection techniques." *Journal of Network and Computer Applications*, 60, 19–31.
13. Ahmed, M., Mahmood, A. N., & Hu, J. (2016). "A survey of network anomaly detection techniques." *Journal of Network and Computer Applications*, 60, 19–31.
14. Mishra, P., Varadharajan, V., Tupakula, U., & Pilli, E. S. (2019). "A detailed investigation and analysis of using machine learning techniques for intrusion detection." IEEE Communications Surveys & Tutorials, 21(1), 686–728.
15. Buczak, A. L., & Guven, E. (2016). "A survey of data mining and machine learning methods for cyber security intrusion detection." IEEE Communications Surveys & Tutorials, 18(2), 1153–1176