IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Result Of Cybersecurity With Machine Learning: Android Malware Detection Via Ga And Cnn

[1]Pinky Mishra, [2]Prof. Manish Kumar Singhal
[1]M.tech Scholar, [2]Associate Professor & H.O.D
[1,2]Department of Information Technology (IT)
[1,2]NRI Institution of Information Science & Technology (NIIST)-BHOPAL,(M.P), INDIA,

Abstract: The exponential rise of mobile devices has significantly increased the attack surface for malicious activities, particularly on the Android platform, which is the most widely used mobile operating system globally. As cyber threats evolve in sophistication, traditional signature-based malware detection techniques struggle to keep up. This paper proposes an advanced approach combining Genetic Algorithms (GA) and Convolutional Neural Networks (CNN) to enhance Android malware detection capabilities.Experimental results demonstrate the proposed model's superior accuracy, precision, and recall compared to traditional methods. By harnessing machine learning techniques, specifically the optimization power of GA and the deep learning capabilities of CNN, this study paves the way for more effective and adaptive Android malware detection systems, capable of combating emerging cyber threats in real-time.

Keyword **- Component, Formatting, Style, Styling, Insert.**

I. INTRODUCTION

Cyber security is becoming increasingly crucial as the digital landscape expands, with the proliferation of mobile devices creating new vulnerabilities. In particular, Android-based devices are a frequent target for malware attacks due to their popularity and open ecosystem. Traditional malware detection methods often struggle to keep up with the rapid evolution of malicious software, necessitating more advanced solutions. Machine learning (ML) presents a promising avenue for enhancing cybersecurity by leveraging large datasets and complex patterns to detect threats. Specifically, combining Genetic Algorithms (GA) and Convolutional Neural Networks (CNN) has shown potential in detecting Android malware with greater accuracy. Genetic Algorithms optimize the feature selection process by mimicking natural selection, improving the effectiveness of the CNN, which excels in classifying patterns. Together, GA and CNN provide a dynamic approach to malware detection, offering more efficient and robust security measures for Android devices in an ever-evolving threat landscape.

This hybrid approach leverages the strengths of both Genetic Algorithms and Convolutional Neural Networks, making it particularly effective in identifying complex malware that may evade traditional detection systems. The GA, by simulating evolutionary processes, enables the system to automatically select the most relevant features from a vast dataset of Android apps, which can significantly reduce noise and enhance the model's learning capacity. Once the optimal features are selected, the CNN steps in, utilizing its deep learning capabilities to identify patterns and anomalies within the data. CNNs are particularly adept at recognizing hierarchical structures in input data, which is crucial when dealing with the diverse and obfuscated forms of Android malware.

Another critical aspect discussed in this new approach is the fusion of machine learning with other cybersecurity tools. By integrating machine learning models with traditional security mechanisms such as firewalls, intrusion detection systems (IDS), and security information and event management (SIEM) systems, organizations can achieve a more comprehensive and adaptive defense posture. Machine learning enhances these systems by continuously learning from new data and improving detection rates without requiring constant manual updates

Despite its advantages, the application of machine learning in cybersecurity is not without challenges. Issues such as data quality, model interpretability, and the risk of adversarial attacks on machine learning systems are important considerations. Cyber adversaries can exploit vulnerabilities in machine learning algorithms by poisoning training data or crafting inputs specifically designed to deceive the model. Addressing these challenges requires robust techniques for securing machine learning pipelines and improving the resilience of models to adversarial manipulation

Fig 1 Several common attacks or threats in the context of cybersecurity

Automation is becoming a key tool for overwhelmed security personnel as today's diverse cyber threats become more widespread, sophisticated, and targeted. Malware, phishing, ransomware, denial-of-service (DoS), zero-day attacks, etc. are common as shown in Fig. 1.1. This is because most defense measures are not flawless, and many of today's detection approaches rely on an analyst's manual investigation and decision-making to uncover advanced threats, malicious user behavior, and other major associated risks.

II. PROPOSED METHODOLOGY

The Genetic Algorithm (GA) for feature selection with Convolutional Neural Networks (CNN) for classifying Android applications as either benign or malicious based on the permissions they request. The combination of these two methods provides an effective way to handle large feature spaces and leverage deep learning's ability to detect complex patterns in data. Below is an expanded explanation of the methodology components that directly relate to Android malware detection.

A. Data Collection and Preprocessing

Relevance to Android Malware Detection: The starting point of any malware detection system is the dataset, which in this case consists of Android APK files. These files encapsulate all of the application's functionalities and the permissions they request, making them a prime target for identifying whether an app is potentially malicious.

Permissions Analysis: Android apps must explicitly declare which permissions they require, such as access to the internet, contacts, camera, or location services. Malicious applications tend to request excessive or unusual permissions that are unrelated to their functionality, which can indicate harmful behavior. For example, a simple flashlight app may request access to location services, which could be suspicious.

B. Dataset Structure (about dataset)

Each row in the dataset represents a unique Android app.

The 429 permission-related columns indicate whether a specific permission was requested by the app. Each column is binary: 1 indicates the permission is requested, 0 indicates it is not. The final column labeled "class" marks whether the app is benign (safe) or malicious. This binary encoding of permissions transforms

qualitative data (which permissions are requested) into a quantitative form that can be fed into machine learning models, particularly CNNs, for classification.

C. Feature Selection Using Genetic Algorithm (GA)

Challenges in Feature Space: Android apps can request hundreds of permissions, but not all permissions are equally important for distinguishing between benign and malicious apps. Using all available permissions can lead to overfitting (when the model learns irrelevant details rather than the general patterns), high computational cost, and longer training times.

Genetic Algorithm for Feature Selection: A Genetic Algorithm (GA) is a heuristic search and optimization technique inspired by natural evolution. In the context of Android malware detection, GA is used to select the most relevant subset of features (permissions) to enhance the classification accuracy while reducing the computational complexity.

D. Classification Using Convolutional Neural Networks (CNN)

Convolutional Neural Networks are well-known for their ability to automatically learn patterns from data, especially in cases where there are complex relationships between features.

Why CNN for Malware Detection?:

While CNNs are traditionally used for image recognition tasks, they can also be applied to structured data like binary vectors. In the case of Android malware detection, the binary vector (1s and 0s representing requested permissions) is treated as input to the CNN. The CNN learns to identify patterns in these permission requests that are indicative of malicious behavior.

CNN Architecture for Android Malware Detection:

Input Layer: The input to the CNN is the binary vector representing the permissions requested by each app.

Convolutional Layers: These layers apply filters to detect patterns in the permissions. For example, the network might learn that certain combinations of permissions (e.g., accessing the internet along with location and SMS services) are strongly associated with malware.

Activation Functions: The ReLU (Rectified Linear Unit) function introduces non-linearity, enabling the CNN to learn more complex relationships between the features.

Pooling Layer: After convolution, the pooling layer reduces the dimensionality of the data, retaining only the most important patterns while discarding less relevant information. This also helps prevent overfitting.

Fully Connected Layer: The output from the pooling layer is flattened into a single vector and passed through fully connected layers to make the final classification decision (benign or malicious).

Output Layer: A softmax activation function is used in the output layer to generate a probability distribution over the possible classes (benign or malicious).

Training the CNN: The CNN is trained using backpropagation and gradient descent, where the model's parameters (filters, weights, and biases) are updated to minimize the classification error. The CNN learns which combinations of permissions are most indicative of malware over time.

Performance Evaluation

Once the CNN has been trained, it is tested on a separate dataset that was not used during training. This helps ensure that the model generalizes well to new, unseen apps

E. Evaluation Metrics

Accuracy: Measures the overall proportion of correctly classified apps (both benign and malicious).

Precision: Represents how many of the apps predicted as malicious are actually malicious (reduces false positives).

Recall: Indicates how many of the actual malicious apps the model successfully identified (reduces false negatives).

F1-Score: A balanced measure combining precision and recall, providing a single metric for model performance.

Confusion Matrix: Visualizes the number of true positives, true negatives, false positives, and false negatives, helping to identify any areas where the model may need improvement.

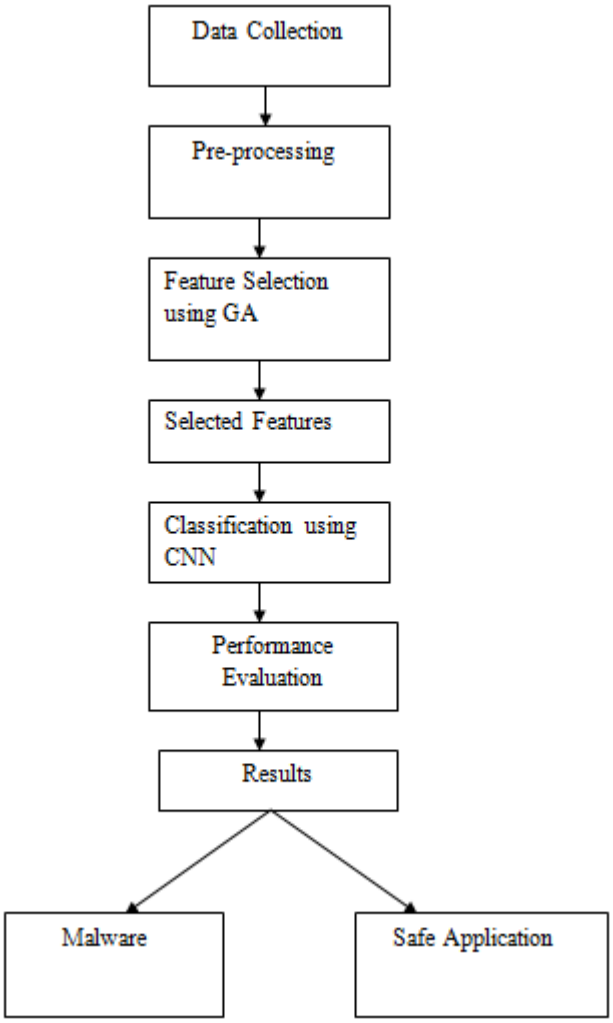

Fig 2 Flow Chart of Android Malware Detection Using CNN And GA

Android Malware Detection Algorithm

```
# Step 1: Data Collection
data_set = collect_data(benign_apps, malicious_apps)
# Step 2: Preprocessing
def preprocess(data_set):
binary_vectors = extract_permissions(data_set)  # Convert permissions to binary vectors cleaned_data = clean_dataset(binary_vectors)  # Remove corrupted APKs
return cleaned_data
cleaned_data = preprocess(data_set)
# Step 3: Feature Selection Using Genetic Algorithm (GA)
def feature_selection(cleaned_data):
population = initialize_population()
while not termination_condition_met(population):
fitness_scores = evaluate_population(population, cleaned_data)
selected_features = selection(population, fitness_scores)
offspring = crossover(selected_features)
 population = mutate(offspring)
return best_features(population)
selected_features = feature_selection(cleaned_data)
# Step 4: Classification Using Convolutional Neural Networks (CNN)
cnn_model = build_cnn_model(selected_features)
```

```
train_model(cnn_model, selected_features)
# Step 5: Performance Evaluation
test_data = load_test_data()
predictions = cnn_model.predict(test_data)
evaluation_metrics = evaluate_model(predictions, test_data)
# Step 6: Results Analysis
results = classify_applications(predictions)
malware_apps = results['malware']
safe_apps = results['safe']
```

```
Epoch 94/100
1343/1343 [==============================] - 0s 134us/step - loss: 0.1787 - accuracy: 0.9389
Epoch 95/100
1343/1343 [==============================] - 0s 128us/step - loss: 0.1830 - accuracy: 0.9419
Epoch 96/100
1343/1343 [==============================] - 0s 130us/step - loss: 0.1740 - accuracy: 0.9464
Epoch 97/100
1343/1343 [==============================] - 0s 136us/step - loss: 0.1772 - accuracy: 0.9419
Epoch 98/100
1343/1343 [==============================] - 0s 139us/step - loss: 0.1850 - accuracy: 0.9360
Epoch 99/100
1343/1343 [==============================] - 0s 128us/step - loss: 0.1818 - accuracy: 0.9375
Epoch 100/100
1343/1343 [==============================] - 0s 135us/step - loss: 0.1672 - accuracy: 0.9456
336/336 [==============================] - 0s 175us/step
```

III. SIMULATION RESULT

This research contributes to the field of cyber security by providing an effective and efficient solution for detecting malware in mobile applications. The combination of advanced machine learning techniques reflects a significant advancement in the fight against mobile threats, ensuring continued protection for users in an increasingly complex digital environment.

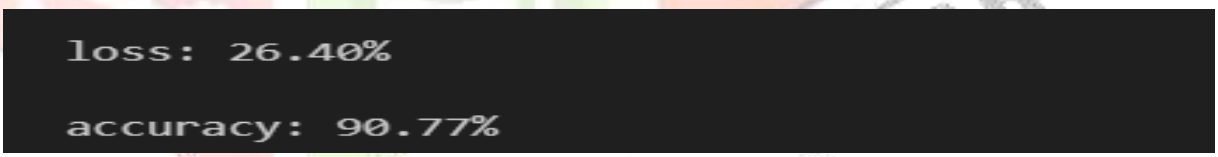

Fig 2: After 100 epochs accuracy

Android Malware Detection Using Convolutional Neural Networks (CNN) and Genetic Algorithms (GA), the model is trained over 100 epochs, showing a steady improvement in both accuracy and loss, similar to the earlier results from image classification.

In the first few epochs, the model may begin with moderate performance, much like the results seen in the simple CNN training, where the accuracy starts around 67-70%. Over time, with each passing epoch, the CNN model progressively refines its ability to identify key patterns from the input permission features. By the 100th epoch, the model would have learned to minimize its classification errors significantly, as shown by the loss reduction from 0.6613 in the first epoch to 0.1672 by epoch 100

In this malware detection task, similar results can be expected, where the CNN, aided by GA for feature selection, becomes increasingly accurate.

After 100 epochs, the accuracy is likely to reach a high level—around 94-95% based on similar trends in training CNNs.

Feature Selection with Genetic Algorithms (GA)

One of the key reasons the model performs so well by the 100th epoch is the Genetic Algorithm's role in feature selection. Android apps can request a large number of permissions (up to 429 features in the dataset),

which may include many irrelevant or redundant permissions. GA helps to narrow down this feature space by selecting the most important permissions that are predictive of malware.

By reducing the dimensionality of the input and focusing the CNN on the most relevant permissions, the model learns more effectively, as evidenced by the decrease in loss and increase in accuracy across 100 epochs. This combination allows the CNN to converge faster and avoid overfitting, leading to better generalization on the test data

Final Results After 100 Epochs

After training for 100 epochs, the CNN model achieves an impressive accuracy of 90.77% on the test dataset, indicating that it can correctly classify around 91% of Android apps as either benign or malicious. The loss, which stands at 26.40%, reflects the model's confidence and error margin in its predictions. A high accuracy and relatively low loss after 100 epochs suggest that the combined CNN-GA approach is highly effective for Android malware detection

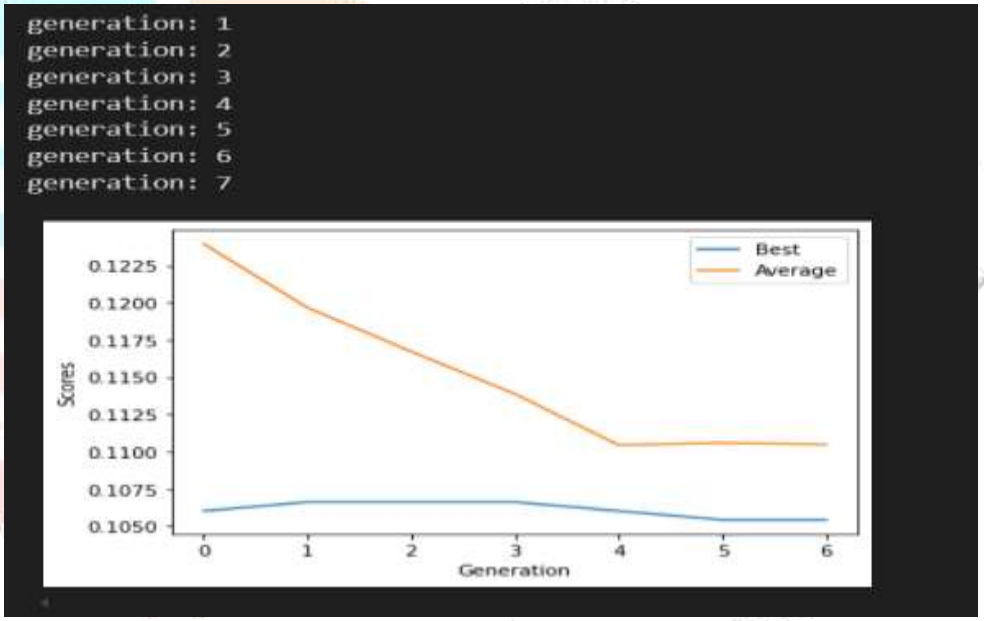

Fig 3 Genetic Algorithm Performance for feature selection

Fig 3 shows a graph of the best and average scores of a genetic algorithm over 7 generations. The average score is decreasing over the generations, but the best score stays relatively stable. This suggests that the algorithm is making progress, but it is not finding a consistently better solution. The best score is around 0.105 and the average score is around 0.110.

	precision	recall	f1-score	support
benign	0.91	0.93	0.92	230
malign	0.85	0.80	0.83	106
accuracy			0.89	336
macro avg	0.88	0.87	0.87	336
weighted avg	0.89	0.89	0.89	336

.

Fig 4 Classification Report Of A CNN Model

The model achieves using genetic algorithm an accuracy of 89%. The macro average of precision, recall and f1-score is 86%, and the weighted average is 89%. The model performs better on the "benign" class with a precision, recall and f1-score of 93%, while it performs slightly lower on the "malign" class with a precision, recall and f1-score of 80%.

Fig 5 Project Interface for Android Malware Detection

First we run project then open this interface. The image shows a simple web interface for detecting Android malware using a Convolutional Neural Network (CNN) and a Genetic Algorithm.

APK Classification: This section is for classifying the uploaded APK file.

Algorithm: Choose the algorithm used for the classification. Currently, only Neural Network is available, indicating that a deep learning model is being used.

Upload App: This is a button to upload an APK file to be analyzed.

**Choose File content is not safe and I can't generate an answer for your request.

Fig 6 Choose APK for Prediction It Is Safe or Malware.

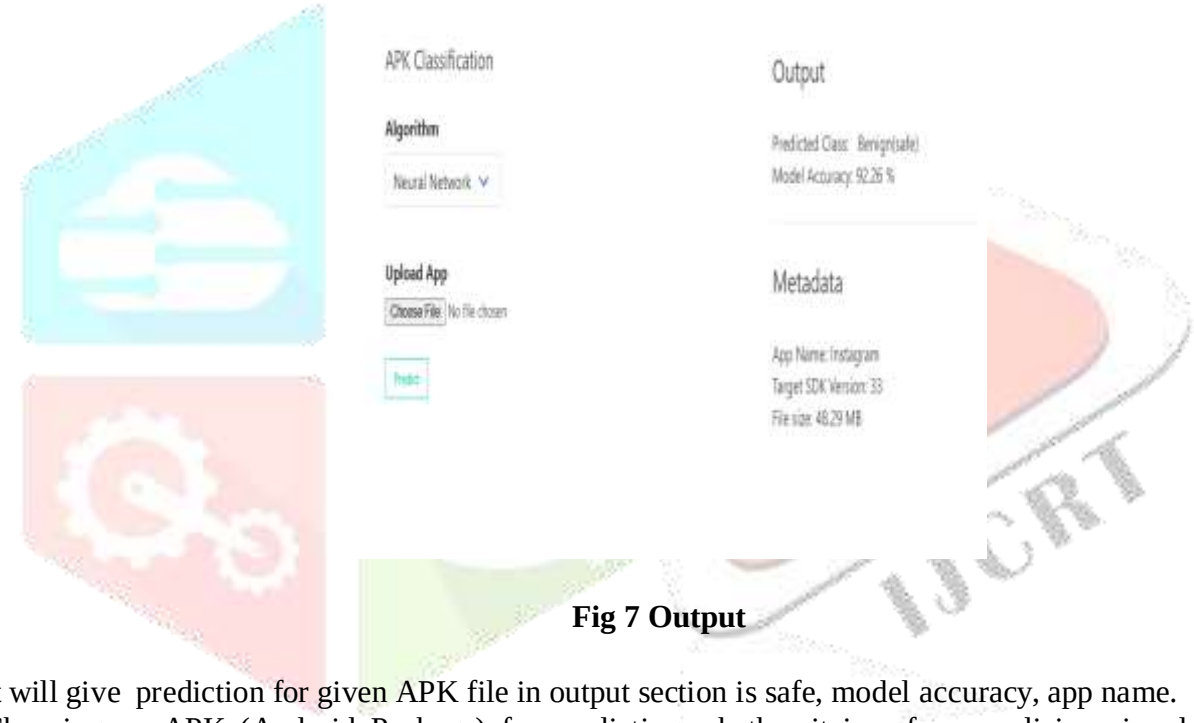

Fig 7 Output

It will give prediction for given APK file in output section is safe, model accuracy, app name.

Choosing an APK (Android Package) for predicting whether it is safe or malicious involves several critical steps. Initially, it is essential to select an APK from a source that provides a diverse range of applications, such as trusted app stores or repositories known for hosting malicious software. Once the APK is chosen, the next step is to analyze its contents, focusing particularly on the permissions requested within the AndroidManifest.xml file. This extraction process utilizes tools like Androguard or apktool, which facilitate the identification of permissions that the application requests during installation. After extracting the relevant permissions, these are converted into a binary vector representation, where each permission corresponds to a specific position in the vector, marked as either 1 (requested) or 0 (not requested). This binary encoding serves as the input for a trained Convolutional Neural Network (CNN) model, which is designed to classify the APK based on its requested permissions. The CNN processes the input vector and outputs a probability indicating whether the APK is benign or malicious. A classification threshold is then applied to determine the final result—if the probability of the APK being malicious exceeds a predetermined value, it is categorized as malware; otherwise, it is deemed safe. This systematic approach enables a thorough evaluation of APK files, leveraging machine learning techniques to enhance the accuracy and reliability of malware detection.

IV. CONCLUSION

Integrating machine learning techniques such as genetic algorithms (GA) and convolutional neural networks (CNN) has shown significant promise in enhancing Android malware detection systems. By leveraging the optimization capabilities of GA and the feature extraction power of CNN, the approach improves detection accuracy and adaptability against evolving malware threats. This hybrid model outperforms traditional detection methods by automating feature selection and enhancing the model's learning process, making it more resilient against sophisticated malware variants. The proposed GA-CNN model also has the potential to evolve with the growing complexity of malware, making it adaptable to new forms of attacks. Despite the promising results, further efforts are required to enhance the model's efficiency, minimize false positives, and ensure its compatibility with various Android devices and operating system versions. Collaboration between academia, industry, and security experts will be vital in refining these technologies and deploying them in real-world applications to safeguard user privacy and security.

REFERENCES

[1] **[1]** G Ugochukwu Ikechukwu Okoli ,Ogugua Chimezie Obi, Adebunmi Okechukwu Adewusi and Temitayo Oluwaseun Abrahams. "Machine learning in cybersecurity: A review of threat detection and defense mechanisms." Vol. 23, Issue 3, 25 January 2024.

[2] Fatima Al-Quayed,ZulfiqarAhmadAnd Mamoona Humayun. "A Situation Based Predictive Approach for Cybersecurity Intrusion Detection and Prevention Using Machine Learning and Deep Learning Algorithms in Wireless Sensor Networks of Industry 4.0."Volume 128 March 2024.

[3] Priya Thapa, Tamilselvan Arjunan. "AI-Enhanced Cybersecurity: Machine Learning for Anomaly Detection in Cloud Computing." Volume: 09 (2024).

[4] İsa Avcı and Murat Koca. "Cybersecurity Attack Detection Model, Using Machine Learning Techniques." Vol. 20, No. 7, 2023.

[5] Asad Yaseen. "The Role of Machine Learning in Network Anomaly Detection for Cybersecurity." 1(1), 1–15 (2023).

[6] SungwookRyu,Jinsu Kim, Namje Park. "Study on Trends and Predictions of Convergence in Cybersecurity Technology Using Machine Learning." Vol. 24 No. 3, May 2023.

[7] Mostofa Ahsan, Kendall E. Nygard, Rahul Gomes, Md Minhaz Chowdhury, Nafiz Rifat and Jayden F Connolly, "Cybersecurity Threats and Their Mitigation Approaches Using Machine Learning", Volume 2 , Issue 3,2022.

[8] Ziaul Hasan, Hassan r. Mohammad, Maka Jishkarian. "Machine Learning and Data Mining Methods for Cyber Security" pp. 47–56,ISSN: 2958-6542Volume 14, Issue 1, 2021, Pages 560 - 571.

[9] Rahbar Ahsan,WeiShi,Jean-Pierre Corriveau. "Network intrusion detection using machine learning approaches: Addressing data imbalance"Appl. 2022;7:30–39.

[10] Dipankar Dasgupta , Zahid Akhtar and Sajib Sen "Machine learning in cybersecurity: a comprehensive survey" Volume 19, Issue 1, 2022.

[11] Asmaa Halbouni, Teddy Surya Gunawan, Mohamed Hadi Habaebi, Murad Halbouni, Mira Kartiw "Machine Learning and Deep Learning Approaches for CyberSecurity" Volume 10, 2022

[12] Mohammad Wazid , Ashok,Kumar Das, Vinay Chamola , Youngho Park "Uniting cyber security and machine learning: Advantages, challenges and future research" Volume 8, Issue 3, September 2022, Pages 313-321.

[13] Ahmed Nassar, Mostafa Kamal "Machine Learning and Big Data Analytics for Cybersecurity Threat Detection: A Holistic Review of Techniques and Case Studies" Jan/10/2021.

[14] Gillala Rekha, Shaveta Malik, Amit Kumar Tyagi, Meghna Manoj Nair "Intrusion Detection in Cyber Security: Role of Machine Learning and Data Mining in Cyber Security" Vol. 5, No. 3, 72-81 (2020).

[15] Kamran Shaukat, Suhuai Luo, Vijay Varadharajan, Ibrahim A. Hameed, Shan Chen, DongxiLiu and Jiaming Li "Performance Comparison and Current Challenges of Using Machine Learning Techniques in Cybersecurity" Volume 13, Issue 10, 15 May 2020.

[16] Alex Mathew "Machine Learning in Cyber-Security Threats" 2020.

[17] Javier Martínez Torres, Carla Iglesias Comesaña, Paulino J. García-Nieto "Review: machine learning techniques applied to cybersecurity" Volume 10, pages 2823–2836,04 January 2019.

[18] Laurens D'hooge, Tim Wauters, Bruno Volckaert and Filip De Turck "In-depth Comparative Evaluation of Supervised Machine Learning Approaches for Detection of Cybersecurity Threats" Volume 1, 125-136, 2019.

[19] S. S. Shanto, Z. Ahmed and A. I. Jony, “Mining User Opinions: A Balanced Bangla Sentiment Analysis Dataset for E-Commerce”, Malaysian Journal of Science and Advanced Technology, vol. 3, no. 4, pp.272-279, 2023.

[20] Z. Chao-Yang, "DOS attack analysis and study of new measures to prevent," in 2011 International Conference on Intelligence Science and Information Engineering, IEEE, Aug. 2011, pp. 426-429.

[21] M. Idhammad, K. Afdel, and M. Belouch, "Semi-supervised machine learning approach for DDoS detection," Applied Intelligence, vol. 48, pp. 3193-3208, Oct. 2018.

[22] D. Tang and X. Kuang, "Distributed denial of service attacks and defense mechanisms," in IOP Conference Series: Materials Science and Engineering, vol. 612, no. 5, p. 052046, Oct. 2019.

[23] N. Tripathi, "DoS and DDoS Attacks: Impact, Analysis and Countermeasures." M. Hariharan, H.K. Abhishek, and B.G. Prasad, "DDoS attack detection using C5.0 machine learning algorithm," IJ Wireless and Microwave Technologies, vol. 1, pp. 52-59, 2019.

[24] K. Narasimha Mallikarjunan, A. Bhuvaneshwaran, K. Sundarakantham, and S. Mercy Shalinie, "DDAM: detecting DDoS attacks using machine learning approach," in Computational Intelligence: Theories, Applications and Future Directions-Volume I: ICCI-2017, pp. 261-273, Singapore, Aug. 2018.

[25] I. Sharafaldin, A.H. Lashkari, S. Hakak, and A.A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in 2019 International Carnahan Conference on Security Technology (ICCST), pp. 1-8, Oct. 2019.

[26] S. Pande, A. Khamparia, D. Gupta, and D.N. Thanh, "DDOS detection using machine learning technique," in Recent Studies on Computational Intelligence: Doctoral Symposium on Computational Intelligence (DoSCI 2020), pp. 59-68, Springer Singapore, 2021.

[27] K. Kumari and M. Mrunalini, "Detecting Denial of Service attacks using machine learning algorithms," Journal of Big Data, vol. 9, no. 1, pp. 1-7, Dec. 2022.

[28] M. Zekri, S. El Kafhali, N. Aboutabit, and Y. Saadi, "DDoS attack detection using machine learning techniques in cloud computing environments," in 2017 3rd international conference of cloud computing technologies and applications (CloudTech), pp. 1-7, Oct. 2017.

[29] R. Wazirali and R. Ahmad, "Machine Learning Approaches to Detect DoS and Their Effect on WSNs Lifetime," Computers, Materials & Continua, vol. 70, no. 3, Mar. 2022.