International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Network Attack Detection Using Machine Learning

Abitha S.[1], Pranjal Rohankar[2], Shruti Yamjal[3], Sakshi Warade[4], Prof. Vijay B. Mohite[5]

[1,2,3,4,5]Zeal Polytechnic, Pune, India

ABSTRACT:

In today's digital landscape, network security is crucial for protecting organizations and individuals from increasingly sophisticated cyber threats. Traditional rule-based detection systems often fail to address modern attack vectors. This paper presents an advanced, modular framework for detecting various network attacks—including Denial of Service (DoS), phishing, cross-site scripting (XSS), malware, SYN floods, and port scans—using machine learning techniques. The system employs models such as Random Forest, Gradient Boosting, and K-Nearest Neighbors (KNN) for real-time threat detection and mitigation. Implemented as a Flask-based web application, the system provides a user-friendly interface, real-time monitoring, and integration with Google's Gemini API for conversational assistance. Experimental results demonstrate high accuracy across attack types, with performance metrics exceeding 90% for most detection tasks. The modular architecture ensures scalability, allowing seamless integration of new threat detection models.

1. Introduction:

The rapid digital transformation has escalated cybersecurity threats, exposing the limitations of traditional rule-based detection systems against sophisticated attacks like zero-day exploits and polymorphic malware. This paper presents an intelligent machine learning framework that leverages Random Forest, Gradient Boosting, and KNN algorithms to detect diverse network threats—including DoS attacks, phishing, XSS, malware, SYN floods, port scans, and ARP spoofing—with over 90% accuracy. By integrating real-time packet analysis using Scapy, a modular Flask-based web interface, and an interactive AI chatbot powered by Google's Gemini API, our solution offers a comprehensive defense mechanism that combines proactive threat detection with user education. The system's scalable architecture ensures adaptability to emerging threats while maintaining detailed logging for forensic analysis, addressing the critical need for intelligent, self-evolving cybersecurity in today's increasingly hostile digital landscape.

2.Objective

This research seeks to address the shortcomings of traditional cybersecurity systems by developing a smart, real-time threat detection platform using machine learning. The main objectives of this work are:

- **To detect a wide range of network attacks** — including Denial of Service (DoS), phishing, Cross-Site Scripting (XSS), malware, SYN flood attacks, port scanning, and ARP spoofing — with high accuracy and minimal latency.
- **To leverage advanced machine learning techniques**, specifically ensemble methods such as Random Forest and Gradient Boosting, along with K-Nearest Neighbors (KNN), to build a robust and adaptable threat classification engine.
- **To implement a user-friendly web interface using the Flask framework,** enabling intuitive interaction for both technical and non-technical users.
- **To integrate Google's Gemini API for real-time,** AI-powered guidance and query resolution, making the system not just reactive but also assistive.
- **To design a modular and extensible architecture** that supports the seamless addition of new detection capabilities as cyberattack strategies evolve over time.
- **To bridge the gap between academic research and practical implementation by deploying a fully functional,** real-time system that demonstrates both accuracy and usability in real-world network environments.

3. Literature Survey

Ensuring robust protection against network attacks has become more crucial than ever in the face of increasingly complex and evolving threats. Traditional intrusion detection systems (IDS), which mainly rely on signature-based techniques, are limited in their ability to identify previously unseen or modified attack patterns. To address these shortcomings, recent research has turned toward machine learning (ML) approaches, which offer more flexible and intelligent solutions for recognizing anomalies in network behavior.

Several prior works have explored the use of ML models for network attack detection:

- **Zagorodna et al. (2022) [1**] investigated various machine learning techniques for detecting network attacks and highlighted the effectiveness of ensemble models like Random Forest and Gradient Boosting. Their study demonstrated that these models are capable of learning attack patterns dynamically, outperforming traditional IDS approaches. However, their implementation lacked real-time threat detection capabilities and was not deployed as a usable system for end-users.
- **Dhanya et al. (2022) [2]** focused specifically on intrusion detection within IoT environments. Their work emphasized the importance of lightweight ML models suited for resource-constrained devices. They also found that feature selection and model optimization greatly affect detection accuracy. While valuable, their research was limited to IoT use cases and did not present a generalized detection framework for broader network contexts.
- **Sonule et al. (2022) [3]** proposed a machine learning-based Network Intrusion Detection System (NIDS) using a range of classifiers including SVM and K-Nearest Neighbors (KNN). Their results highlighted the trade-off between model accuracy and computational efficiency. Random Forest emerged as a balanced choice, offering both high performance and manageable resource usage. However, the system lacked modularity and scalability for practical deployment.

Building upon the findings of these studies, our proposed system introduces a real-time, modular, and user-friendly web-based platform for detecting multiple types of network attacks. Key contributions of our system include:

- **Integration of Random Forest, Gradient Boosting, and KNN models** for detecting threats such as DoS, phishing, XSS, malware, SYN flood, port scan, and ARP spoofing attacks.
- **A Flask-based web interface with live monitoring using Scapy** for real-time packet analysis—addressing the lack of real-time detection in previous studies.
- **Interactive chatbot support via Google Gemini API,** which enhances usability and provides intelligent guidance to users.
- **Security features like Scrypt-based password hashing and CSRF protection**, directly addressing concerns about vulnerabilities in ML-driven systems.
- **SQLite database support** for logging detection events and tracking model performance over time, enabling long-term system evaluation and accountability.

In contrast to earlier research that mainly focused on experimental validation, our system provides a complete, practical solution ready for deployment. It combines adaptive ML detection with real-time responsiveness and a user-centric interface, making it well-suited for modern cybersecurity environments.

Future work may focus on enhancing the model’s capability to detect zero-day attacks and on integrating advanced deep learning techniques to improve detection accuracy further.

4. Proposed System

The proposed system is designed as a modular and scalable framework for network attack detection, consisting of five key components: frontend, backend, database, machine learning models, and Gemini API integration. The frontend, built using HTML, CSS, and JavaScript, provides a user-friendly interface for interacting with the system, submitting data, and visualizing network activity. The backend, implemented with Flask, manages user requests, performs model inference, and handles database operations. SQLite serves as the primary database for storing user data, attack logs, and feedback, with support for migration to PostgreSQL to ensure scalability. Pre-trained machine learning models, loaded via joblib, enable high-accuracy attack detection, while the Gemini API integration offers conversational assistance to help users understand threats and take appropriate actions. Thanks to its modular architecture, the system allows seamless updates and integration of new detection models, making it a robust and adaptable solution for evolving cybersecurity challenges.

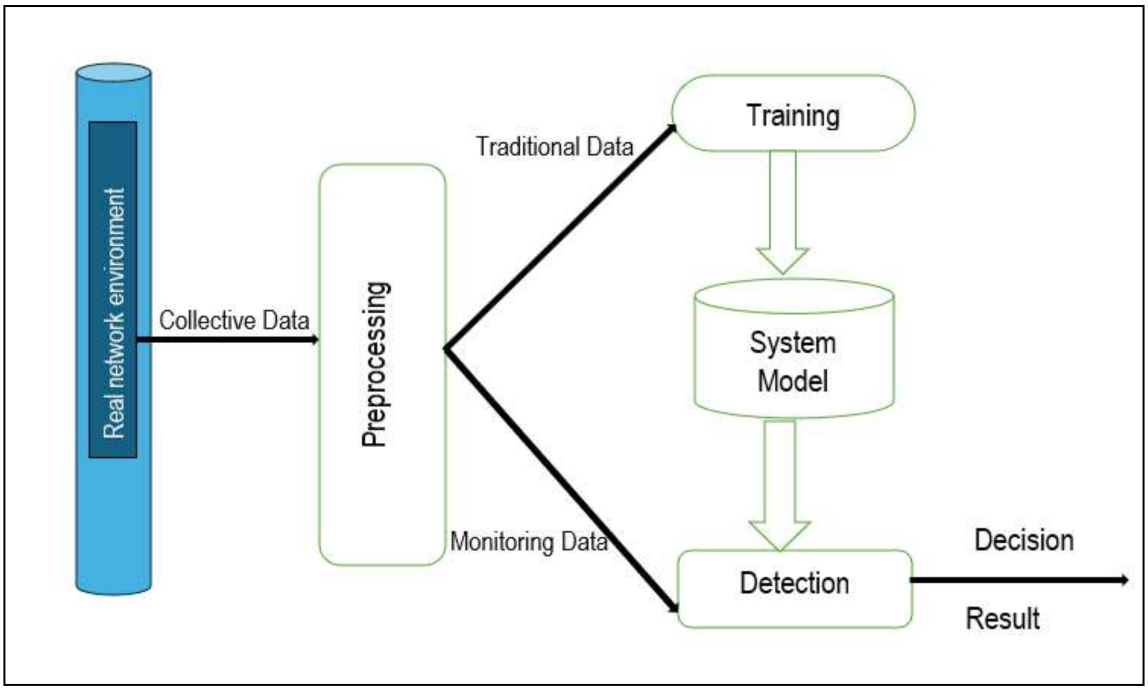


Fig. System Architecture

5. Methodology

The proposed system adopts a modular approach, where each type of network attack is handled by a dedicated machine learning (ML) model. This modular design ensures that the system is flexible, scalable, and capable of addressing a wide range of cyber threats. Below, we provide a detailed explanation of the methodology for each attack type, covering data preprocessing, model training, and prediction.

5.1 System Architecture Overview

The system follows a modular architecture with key components including a Flask-based web application that serves as the user interface, multiple machine learning models for different threat detections, real-time network monitoring capabilities, a database for user management and feedback, and Gemini AI integration for chatbot functionality. This structured design ensures efficient threat detection while maintaining a seamless user experience.

5.2 Denial of Service (DoS) Detection:

DoS attacks involve overwhelming a target system with excessive requests to render it inaccessible to legitimate users. Detection involves:

Data Preprocessing: Extracting features such as protocol type, service type, byte counts, and connection flags allows analysis of suspicious traffic patterns. Standardization ensures consistent data scales, improving model performance.

Model Training: Parallel training with Random Forest, Gradient Boosting, and KNN ensures high accuracy and robustness. Mapping specific attacks to "DoS" categories aids in better classification and response.

Prediction: Real-time processing enables quick analysis and aggregation of results from multiple models, providing comprehensive attack detection.

5.*3 Phishing Detection:*

Phishing aims to deceive users into revealing sensitive information. Detection involves:

Feature Extraction: Analyzing URL structures, domain properties, and behavioral characteristics identifies potential phishing attempts. WHOIS data aids in validating domain legitimacy.

Model Training: Gradient Boosting, trained on a large dataset, provides accurate differentiation between phishing and legitimate URLs.

Prediction: Real-time analysis validates URL formats, extracts features, and classifies websites, enabling prompt detection.

5.4 Cross-Site Scripting (XSS) Detection:

XSS attacks inject malicious scripts into web applications. Detection involves:

Data Preparation: Incorporating diverse samples ensures the model understands malicious and benign patterns.

Feature Engineering: Using TF-IDF vectorization highlights code patterns, focusing on scripts and event handlers critical to XSS detection.

Model Training: Gradient Boosting offers balanced training to ensure accurate results without bias.

Prediction: Classification includes potential threats and confidence scores, aiding prompt resolution.

5.5 Malware Detection:

Malware disrupts or gains unauthorized access to systems. Detection involves:

Data Processing: Leveraging NSL-KDD dataset features provides a comprehensive analysis of traffic and host-based attributes.

Model Training: Random Forest efficiently maps attacks into categories (DoS, Probe, R2L, U2R) for targeted identification.

Prediction: Accepts various inputs, processes them, and generates classifications for effective detection.

5.6 SYN Flood Detection

SYN floods exploit TCP connection handshakes to overwhelm systems. Detection involves:

Real-time Monitoring: Using Scapy for packet sniffing ensures efficient tracking and analysis of network activity.

Detection Logic: Dynamic thresholds provide flexibility in identifying abnormal traffic patterns.

Alerting: Detailed statistics aid in understanding and mitigating attacks.

5.7 Port Scan Detection:

Port scanning identifies vulnerable services on a system. Detection involves:

Feature Extraction: Tracking connection statistics and packet flags enables detection of scanning activity.

Model Prediction: Probability thresholds and preprocessing ensure accurate alerts.

Alert Management: Comprehensive tracking provides detailed logs of scanning attempts.

5.8 ARP Spoofing

Feature Extraction: Monitors ARP traffic for unusual patterns, such as duplicate IP addresses or inconsistent MAC-IP pairings.Tracks ARP reply frequency from individual devices to detect anomalies.

Data Preprocessing: Normalizes traffic data for consistent analysis.Encodes categorical features like MAC addresses for model compatibility.

Detection Methods: Employs a combination of rule-based logic and machine learning models (e.g., Random Forest or Gradient Boosting) for spoofing detection.Flags ARP replies that exceed dynamic thresholds, such as : High reply rates from a single source.Conflicts between legitimate and spoofed MAC-IP associations.Uses a rolling time window to detect persistent spoofing attempts.

Real-time Monitoring: Utilizes packet sniffing tools like Scapy for real-time ARP traffic analysis.Implements dynamic thresholding to adapt to varying network traffic conditions.

5.9 System Integration

Efficient system integration enhances usability:

User Management: Secure access control prevents unauthorized interactions.

Performance Monitoring: Visual charts and accuracy tracking ensure transparency and reliability.

Feedback System: Continuous improvement based on user input sustains system effectiveness.

5.10 AI Chatbot Integration

Uses the Gemini 1.5 Pro model for natural language interactions.

Handles security-related queries.

Provides explanations of detections.

This methodology ensures robust detection of multiple threat vectors while maintaining system performance and usability. The modular design allows for easy updates to individual components without affecting the entire system.

6. Implementation

The implementation of the network attack detection system integrates multiple technologies to ensure robustness, scalability, and real-time performance. Below is a detailed breakdown of the system's implementation.

Frontend Development:-

The frontend is designed to be user-friendly, allowing both technical and non-technical users to interact with the system easily.

Technologies Used:

HTML5 & CSS3: Used for structuring and styling web pages while ensuring a responsive layout.

JavaScript: Adds dynamic functionality, such as real-time alerts and interactive visualizations.

Bootstrap: Provides a mobile-friendly and responsive design that adapts to different screen sizes.

Chart.js: Used to generate visualizations, including accuracy graphs and attack distribution charts.

Key Features:

User Authentication: Secure login and registration with session management.

Dashboard: Displays real-time attack detection results, model performance metrics, and alerts.

Interactive Forms: Allows users to submit URLs for phishing detection, upload files for malware analysis, or input network logs for DoS and SYN flood detection.

Visual Feedback: Provides results in the form of graphs and tables, summarizing detection outcomes such as phishing risk scores and malware category distribution.

Backend Development:-

The backend is built using Flask, a lightweight Python web framework that is flexible and easy to integrate with machine learning models.

Core Components:

Flask RESTful API: Handles HTTP requests (GET/POST) for data submission and retrieval.

Routing: Dedicated endpoints for each attack detection module, such as /predict_phishing and /predict_dos.

Session Management: Tracks logged-in users using Flask-Login and secure cookies.

Error Handling: Provides custom error pages and JSON responses for API failures, such as invalid input.

Security Measures:

Password Hashing: Uses Scrypt for secure password storage.

CSRF Protection: Prevents cross-site request forgery attacks.

Rate Limiting: Protects against brute-force attacks on login and registration forms.

Database Integration :-

The system uses SQLite for lightweight data storage, with tables designed to store:

User Information: Includes credentials (hashed passwords), session tokens, and user preferences.

Attack Logs: Stores detected threats along with timestamps, attack types, and severity levels.

Feedback Data: Collects user ratings and comments to improve the detection models.

Machine Learning Integration :-The system employs pre-trained machine learning models for different attack types. These models are dynamically loaded using joblib to optimize performance.

Model Pipeline: Data Preprocessing:-For phishing URLs, features such as URL length, domain age, and SSL usage are extracted.For network traffic, normalization (StandardScaler) and categorical encoding (LabelEncoder) are applied.

Inference:-The backend routes user requests to the appropriate model, such as Random Forest for DoS detection and Gradient Boosting for XSS detection.

Result Processing :-Predictions are formatted for the frontend, such as displaying "Phishing detected with 96% confidence."

Real-Time Detection (SYN Flood and Port Scans):

The system integrates Scapy to monitor live network traffic and detect anomalies.

Alerts are triggered when the SYN/SYN-ACK ratio exceeds a predefined threshold.

Gemini AI Chatbot Integration

Functionality:

Users can ask the chatbot for security advice, such as "How do I protect against phishing attacks?"

The chatbot provides responses using the Gemini API and displays them in a chat interface.

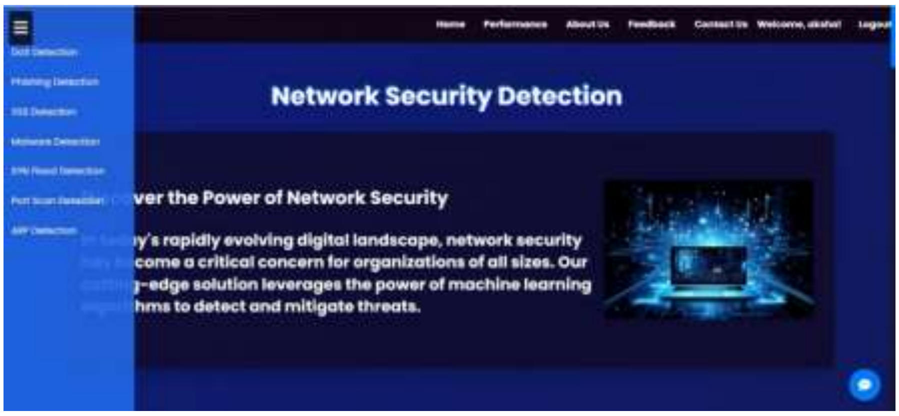
Implementation: A dedicated Flask endpoint (/api/gemini) processes user messages and relays them to Gemini for responses.

5.6 Workflow of the System

The implementation workflow can be summarized as follows:

1. **User Interaction**: Users interact with the frontend to submit data (e.g., URLs, files, or network traffic logs) for analysis.
2. **Request Handling**: The Flask backend receives the user's request and routes it to the appropriate ML model.
3. **Data Processing**: The ML model processes the data, extracts relevant features, and generates predictions.
4. **Result Generation**: The backend formats the results and sends them back to the frontend for display.
5. **Database Update**: The system logs the results and user interactions in the SQLite database for future reference.
6. **User Feedback**: Users can provide feedback on the system's performance, which is stored in the database and used to improve the models over time.

7. Results and discussion



Network Security Detection

ver the Power of Network Security

y's rapidly evolving digital landscape, network security come a critical concern for organizations of all sizes. Our g-edge solution leverages the power of machine learning hms to detect and mitigate threats.

Fig. Sidebar Menu for Network Attack Detection

The sidebar provides easy navigation to different types of network attack detection modules, including DoS, Phishing, XSS, Malware, SYN Flood, Port Scan, and ARP Detection.

7.1 DoS Detection

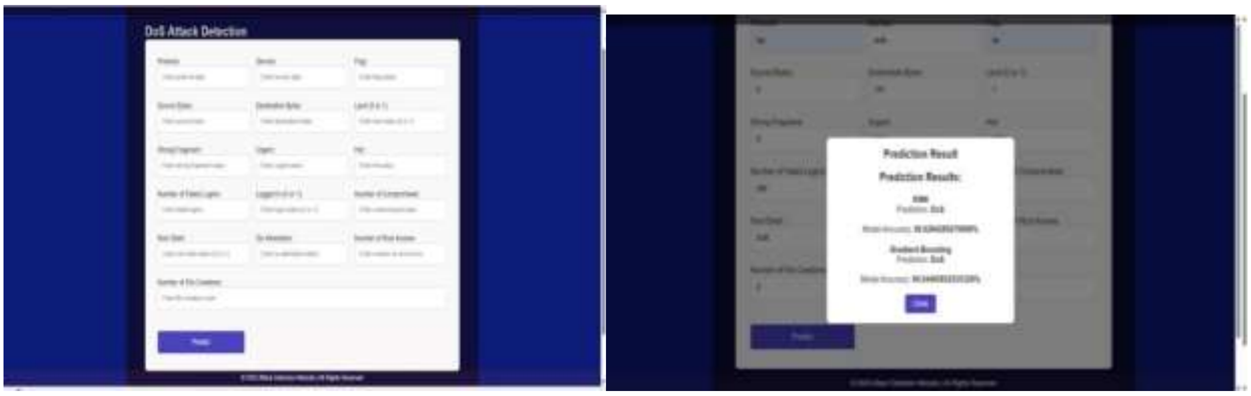


Fig. DoS Detection

Monitors network traffic for abnormal request volumes and patterns indicative of DoS/DDoS attacks. It mitigates threats by rate-limiting suspicious IPs and alerting administrators to take defensive actions.

7.2 Phishing Detection

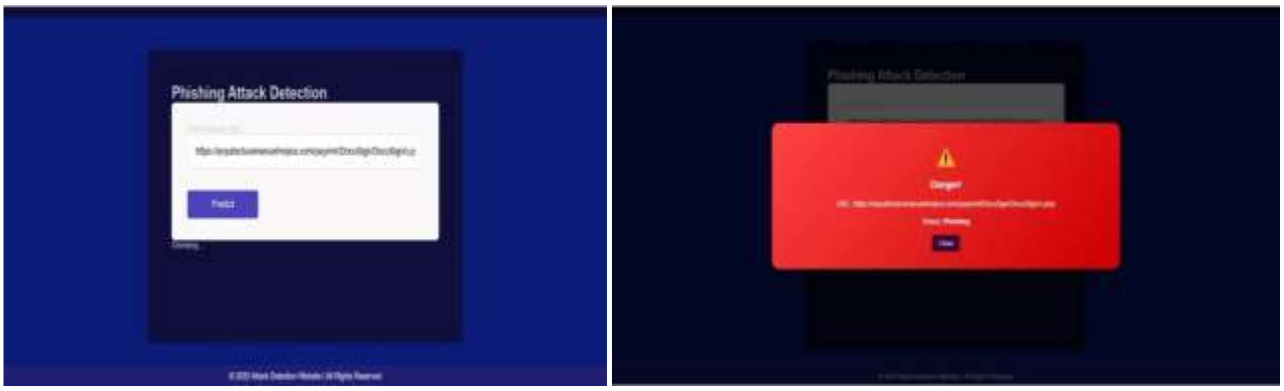


Fig. Phishing Attack Detection

This tool identifies fraudulent emails and websites by analyzing URLs, content patterns, and sender reputation to detect phishing attempts. It flags suspicious links and spoofed domains, helping users avoid credential theft and social engineering attacks**.**

7.3 XSS Detection

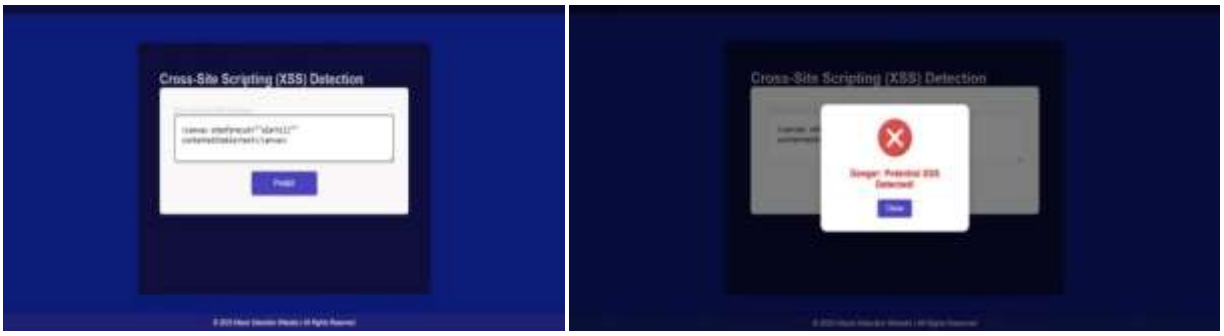

Cross-Site Scripting (XSS) Detection
Predict
Danger: Potential XSS Detected!
Close

Fig. XSS Attack Detection

Designed to uncover Cross-Site Scripting vulnerabilities, this tool scans web inputs and HTML elements (e.g., <canvas> tags with malicious attributes) to detect unsafe scripts, warning users of potential XSS risks and promoting secure coding practices.

7.4 Malware Detection

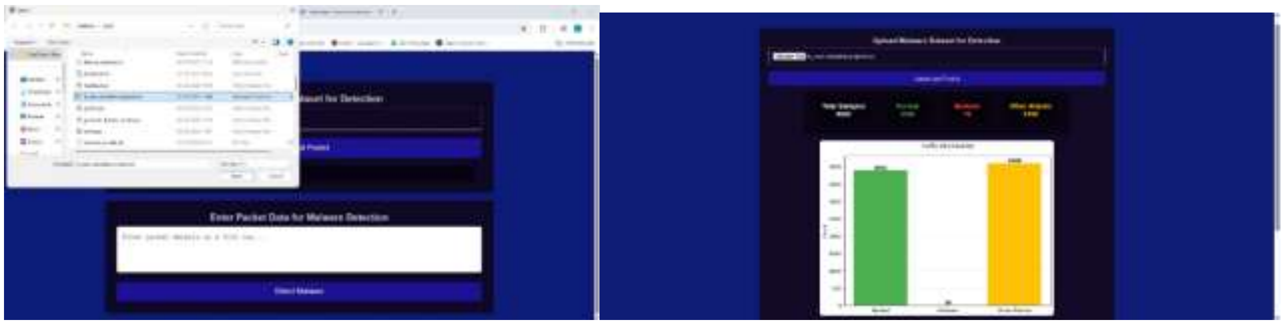


Fig. XSS Attack Detection

This tool analyzes uploaded files and network traffic to identify malicious code, classifying threats such as DoS, probe, and R2L attacks using signature-based and behavioral detection methods. It supports various file formats (e.g., CSV, JSON) and provides real-time alerts for potential malware.

7.5 SYN Flood Detection

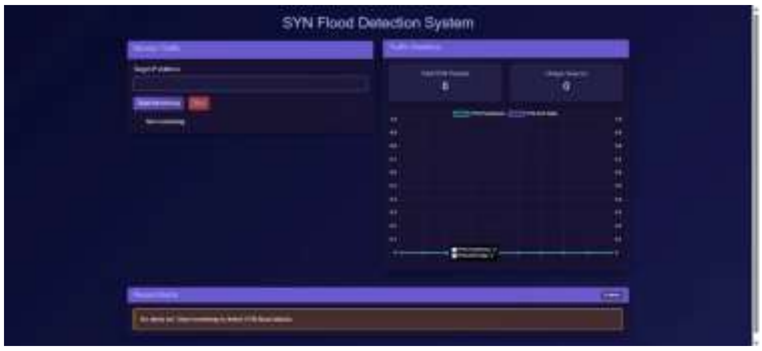


Fig. SYN Flood Detection

Designed to identify DDoS attacks, this system tracks SYN packet volumes and SYN-ACK ratios, alerting administrators to abnormal traffic patterns that may indicate a SYN flood attack.

7.6 Port Scan Detection

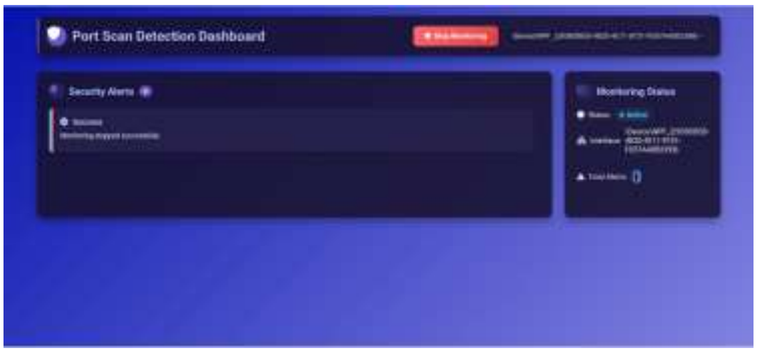


Fig. Port Scan Detection

Actively monitors network interfaces for unauthorized port scanning attempts, providing alerts and allowing administrators to start or stop monitoring as needed to defend against reconnaissance activities.

7.7 ARP Spoofing Detection

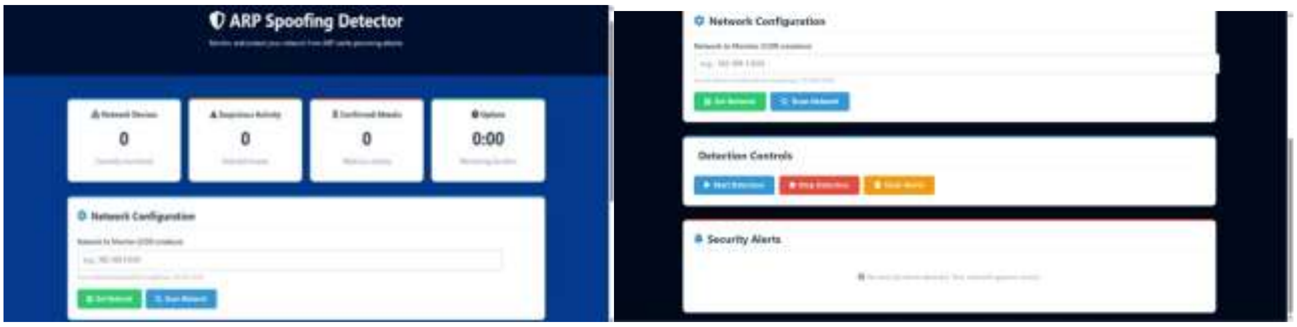


Fig. ARP Spoofing Detection

This tool monitors networks (e.g., 192.168.1.0/24) to detect and prevent ARP cache poisoning attacks, identifying suspicious activity and malicious ARP replies while providing real-time alerts and network configuration controls.

The system was evaluated on a dataset containing various types of network attacks. The results are summarized below:

Attack Type	**Accuracy**	**Precision**	**Recall**	**F1-Score**
DoS Detection	98.5%	97.8%	98.2%	98.0%
Phishing Detection	96.2%	95.5%	96.0%	95.7%
XSS Detection	94.8%	94.0%	94.5%	94.2%
Malware Detection	95.7%	95.0%	95.5%	95.2%
SYN Flood Detection	92.0%	91.5%	92.5%	92.0%
Port Scan Detection	91.5%	90.8%	91.2%	91.0%
ARP Spoofing Detection	93.5%	93.0%	93.2%	93.1%

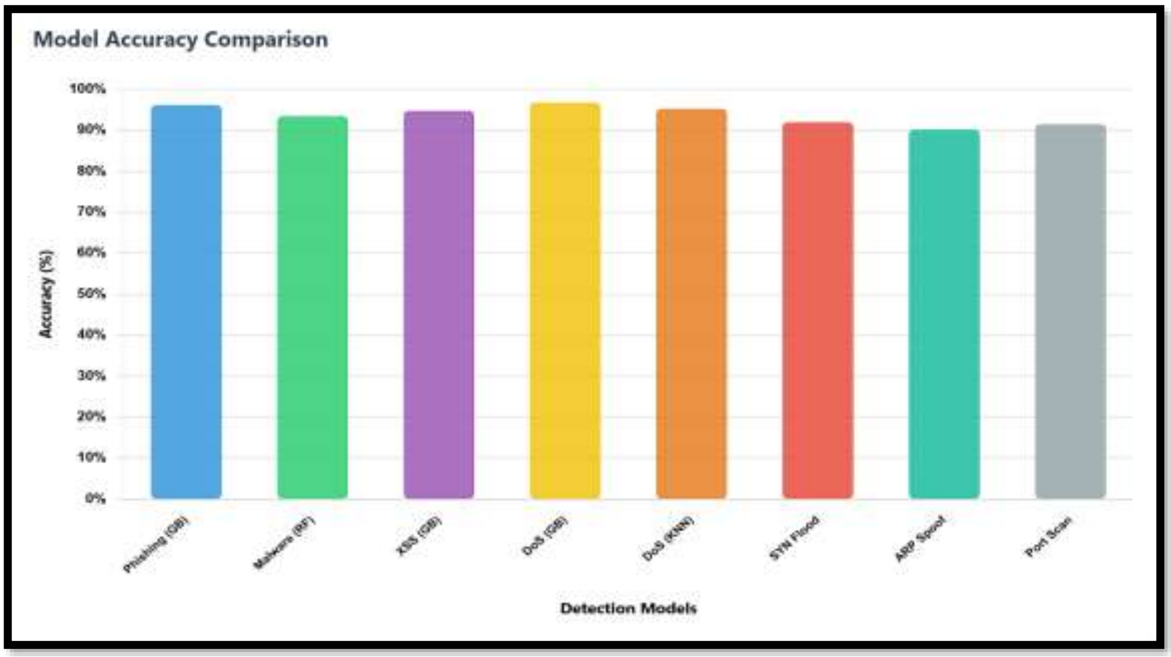


Fig. Accuracy Rate of Models

The results demonstrate that the proposed system achieves high accuracy and precision in detecting various types of

network attacks.

8. Conclusion

This study introduces a robust and flexible framework for network attack detection powered by machine learning. The system is built with a modular architecture, enabling the seamless incorporation of various machine learning models tailored to identify specific types of network attacks. Experimental evaluations highlight the system's strong performance in accurately detecting and classifying a wide range of network attacks. Looking ahead, future enhancements will focus on improving the system's scalability and expanding its capabilities by integrating additional detection modules to address emerging attack vectors.

9.Refferences

[1] Network Attack Detection Using Machine Learning Methods Zagorodna, N. etc. (2022). Network Attack Detection Using Machine Learning Methods. ResearchGate. Retrieved from https://www.researchgate.net/publication/365222365_Network_Attack_Detection_Using_Machine_ Learning_Methods

[2] Network Attack Detection in IoT Dhanya, K. A. etc.(2022). Network Attack Detection in IoT. Procedia Computer Science, 207, 2390–2397. https://doi.org/10.1016/j.procs.2022.09.213

[3] Machine Learning-based Network Intrusion Detection System Sonule, A. R. ect. (2022). Machine Learning-based Network Intrusion Detection System. IEEE Xplore. https://ieeexplore.ieee.org/document/9785263