**Intrusion Detection System for Healthcare Environments: Safeguarding Sensitive Medical Data**

**Introduction**

Healthcare systems are increasingly targeted by cyberattacks due to the sensitive nature of Electronic Health Records (EHRs), patient data, and the critical operations they support. This project presents an **Intrusion Detection System (IDS) for Healthcare Environments**, designed to monitor network traffic in real-time, detect intrusions, and classify them as of March 31, 2025. Tailored for hospitals, clinics, and telemedicine platforms, this IDS integrates a Flask-based backend with a two-stage machine learning pipeline, Scapy for packet capture, and a React frontend styled with Tailwind CSS for user interaction. Its primary goal is to protect healthcare networks from threats while ensuring compliance with regulations like HIPAA.

The system employs a hybrid classification approach: a binary Random Forest classifier identifies normal versus anomalous traffic, followed by an XGBoost multi-class classifier that pinpoints specific attack types relevant to healthcare, such as ransomware or data exfiltration attempts. Real-time alerts are delivered via Flask-SocketIO, with security reinforced by AES-GCM encryption for flow data, JWT-based authentication, and role-based access control (RBAC) to restrict access to healthcare administrators and IT staff. The React frontend, built with Vite, provides a responsive dashboard for monitoring EHR-related traffic, styled with Tailwind CSS for usability in high-stakes environments.

**Tools and Technologies Used**

The IDS leverages a robust stack tailored to healthcare security needs:

**Backend and Server-Side Technologies**

1. **Python** : The core language, chosen for its data science and networking capabilities in healthcare IT.
2. **Flask** : Hosts RESTful APIs (e.g., /start-packet-capture, /predict) for managing healthcare network monitoring.
3. **Flask-S daemon Thread-SocketIO**: Enables real-time alerts (e.g., real-time-data, real-time-alert) for immediate response to threats against EHR systems.
4. **Flask-CORS** : Facilitates secure communication between the backend (port 5000) and frontend (port 5173), adhering to healthcare security policies.
5. **PyMongo**: Interfaces with MongoDB (mongodb://localhost:27017/) to store encrypted user data, activity logs, and audit trails for HIPAA compliance.

**Machine Learning and Data Processing**

1. **Scikit-learn** : Provides Random Forest Classifier for binary detection, preprocessing (StandardScaler), and resampling (SMOTE, RandomUnderSampler) to handle healthcare-specific traffic patterns.
2. **XGBoost**  Implements multi-class classification with optimized hyperparameters to identify healthcare-targeted attacks.
3. **Imbalanced-learn** : Balances datasets with techniques like SMOTEENN, critical for detecting rare healthcare breaches.
4. **Pandas and NumPy** : Manage EHR-related feature extraction and numerical operations.
5. **Joblib** : Serializes models and encoders (e.g., binary\_model.pkl) for deployment in healthcare settings.

**Packet Capture and Security**

1. **Scapy** : Captures network packets from healthcare devices (e.g., EHR servers, medical IoT) using sniff.
2. **PyCrypto (pycryptodome)** : Encrypts flow data with AES-GCM (256-bit key, 96-bit nonce) and uses RSA with PKCS1\_OAEP to protect patient data in transit.
3. **bcrypt** : Secures healthcare staff credentials with salted password hashing.
4. **JWT** : Issues JWTs with a 1-hour expiration for authenticated access by healthcare IT personnel.
5. **psutil** : Monitors server resources, alerting if CPU/memory usage exceeds 85% to ensure system reliability for critical care.

**Frontend Development**

1. **React** : Builds a dynamic UI at http://localhost:5173 for healthcare staff to monitor network activity.
2. **Vite** : A fast build tool for React, ensuring low-latency updates in time-sensitive healthcare environments.
3. **Tailwind CSS** : Styles the frontend with utility classes (e.g., bg-blue-100, text-lg) for a clean, accessible design.

**Miscellaneous**

1. **Logging**: Logs events to app.log for auditability, a key HIPAA requirement.
2. **Threading**: Manages concurrent tasks like packet capture and system monitoring.
3. **pytz** : Formats timestamps in IST (Asia/Kolkata) or configurable healthcare timezone.

**Data Capture Mechanism**

**Tool: Scapy**

**Scapy** drives packet capture, tailored to healthcare networks. The sniff function in capture\_packets intercepts traffic from EHR servers, medical and IoT devices (e.g., MRI machines), and telemedicine endpoints.

**Process**

1. **Endpoint Activation**: The /start-packet-capture POST endpoint accepts an interface (e.g., eth0 for hospital LAN) and spawns a daemon thread.
2. **Packet Sniffing**: sniff(iface=interface, prn=predict\_packet, store=False) captures packets in real-time, optimized for healthcare networks with minimal memory use.
3. **Flow Aggregation**: update\_flow groups packets into flows using (src\_ip, dst\_ip, protocol):
   * Tracks metrics like packet count, bytes, and error rates.
   * Encrypts flow data with AES-GCM to protect patient information, storing it in the flows dictionary with a 10-second timeout.

**Healthcare Context**

In hospitals, Scapy captures traffic from diverse sources—EHR databases, IoT medical devices, and external telemedicine connections—ensuring comprehensive monitoring. Its root-level operation aligns with secure hospital IT setups.

**Capturing Attacks in Healthcare**

The IDS detects healthcare-specific attacks through:

1. **Real-Time Packet Capture**: Scapy sniffs packets from EHR servers and medical devices, identifying anomalies in traffic patterns.
2. **Flow Tracking**: update\_flow aggregates packets, encrypting sensitive data (e.g., patient records in transit) with AES-GCM:
   * **Decryption**: decrypt\_flow\_data ensures secure feature extraction.
3. **Feature Extraction**: extract\_features computes 41 NSL-KDD-aligned features (e.g., src\_bytes, dst\_host\_serror\_rate) from decrypted flows, tailored to detect healthcare breaches.
4. **Healthcare Attack Simulation**: Functions like generate\_real\_time\_data\_neptune (SYN flood to disrupt EHR access) and generate\_real\_time\_data\_back (backdoor to exfiltrate patient data) simulate threats.

**Healthcare Context**

Attacks like ransomware (e.g., warezclient) or data theft (e.g., spy) directly threaten patient care and privacy. Encryption ensures HIPAA-compliant data handling, while simulations mimic real-world healthcare breaches.

**Methods for Identifying Attacks**

The IDS uses a **two-stage classification pipeline** trained on processed\_data.csv, optimized for healthcare threats:

**Stage 1: Binary Classification**

* **Objective**: Distinguish normal healthcare traffic (label 15) from anomalies.
* **Model**: Random Forest Classifier (binary\_model) with 100 estimators and class\_weight='balanced'.
* **Preprocessing**:
  + StandardScaler normalizes features from medical traffic.
  + RandomUnderSampler balances normal EHR access with attack attempts.
* **Training**: Fits on X\_binary and y\_binary (y\_binary = (y != 15).astype(int)).
* **Thresholding**: Uses a 0.5 threshold (overridden from precision-recall optimization) for sensitivity to healthcare threats.
* **Output**: 0 (normal) or 1 (non-normal).

**Stage 2: Multi-Class Classification**

* **Objective**: Identify specific healthcare-relevant attack types.
* **Model**: XGBoost Classifier (multi\_model) with:
  + max\_depth=6, n\_estimators=200, learning\_rate=0.1
  + min\_child\_weight=3, subsample=0.8, colsample\_bytree=0.8
* **Preprocessing**:
  + Non-normal samples are re-encoded with LabelEncoder.
  + RandomUnderSampler limits dominant attacks (e.g., neptune to 20,000); SMOTEENN oversamples rare ones (e.g., worm to 10,000).
  + Weights prioritize healthcare-critical attacks (e.g., warezclient, spy).
* **Training**: Optimizes mlogloss with sample weights.
* **Output**: Maps predictions to label\_to\_name.

**Combined Prediction**

* **Function**: combined\_predict(X, threshold) assigns label 15 to normal traffic or classifies attacks with XGBoost.

**Detectable Attacks in Healthcare**

The IDS detects 33 classes, with emphasis on healthcare threats:

1. **apache2 (0)**: Exploits in hospital web servers.
2. **back (1)**: Backdoors stealing EHR data.
3. **buffer\_overflow (2)**: Overflows in medical software.
4. **ftp\_write (3)**: Unauthorized writes to FTP servers hosting patient files.
5. **guess\_passwd (4)**: Password attacks on EHR logins.
6. **httptunnel (5)**: Tunneling to bypass hospital firewalls.
7. **imap (6)**: Email protocol exploits targeting staff.
8. **ipsweep (7)**: Scans for vulnerable medical devices.
9. **land (8)**: DoS targeting hospital LANs.
10. **loadmodule (9)**: Exploits in medical device firmware.
11. **mailbomb (10)**: Flooding hospital email systems.
12. **mscan (11)**: Multi-protocol scans of healthcare networks.
13. **multihop (12)**: Multi-stage attacks via compromised devices.
14. **neptune (13)**: SYN floods disrupting EHR access.
15. **nmap (14)**: Network reconnaissance in hospitals.
16. **normal (15)**: Legitimate EHR/medical traffic.
17. **perl (16)**: Script exploits in healthcare scripts.
18. **phf (17)**: CGI vulnerabilities in patient portals.
19. **pod (18)**: Ping of Death against medical servers.
20. **portsweep (19)**: Port scans targeting EHR ports.
21. **processtable (20)**: Overflows crashing medical systems.
22. **rootkit (21)**: Privilege escalation on EHR servers.
23. **saint (22)**: Scanning tools targeting healthcare.
24. **satan (23)**: SATAN scans of hospital networks.
25. **smurf (24)**: ICMP amplification disrupting care.
26. **snmpgetattack (25)**: SNMP probes on medical devices.
27. **snmpguess (26)**: SNMP guessing for device access.
28. **spy (27)**: Data exfiltration of patient records.
29. **teardrop (28)**: Fragmented packet DoS on EHR systems.
30. **warezclient (29)**: Ransomware clients encrypting EHRs.
31. **warezmaster (30)**: Ransomware servers in healthcare.
32. **worm (31)**: Worms spreading via medical IoT.
33. **rare (32)**: Rare healthcare-specific attacks.

**Healthcare Context**

Prioritized attacks (e.g., warezclient, spy) reflect ransomware and data theft risks, critical under HIPAA. The two-stage design ensures high recall for rare but severe threats.

**Project Execution**

**Backend Workflow**

1. **Initialization**:
   * Loads models, scalers, and encoders for healthcare traffic analysis.
   * Connects to MongoDB for HIPAA-compliant logging.
2. **Packet Processing**:
   * /start-packet-capture monitors hospital interfaces.
   * predict\_packet processes flows, predicts threats, and emits alerts via SocketIO.
3. **Security**:
   * /login issues JWTs for healthcare staff (e.g., admin for IT, user for clinicians).
   * RBAC restricts access to sensitive endpoints.

**Frontend Workflow**

1. **Setup**:
   * React app at http://localhost:5173 with Vite and Tailwind CSS.
2. **Components**:
   * **Dashboard**: Monitors EHR traffic (src\_ip, prediction).
   * **Alerts**: Displays threats (e.g., “Ransomware Detected: 29-warezclient”).
   * **Admin Panel**: Manages healthcare IT users.

**Additional Features**

* **Performance Monitoring**: Alerts if server resources exceed 85%, critical for uninterrupted patient care.
* **HIPAA Compliance**: Encrypted flows, hashed credentials, and audit logs meet privacy standards.
* **Testing**: Simulates healthcare threats (e.g., ransomware, DoS) for validation.

**Extra Context**

The IDS’s focus on healthcare enhances its relevance in protecting life-critical systems, with potential for integration with medical IoT standards.

**Conclusion**

This IDS for healthcare environments offers a robust, real-time solution to safeguard EHRs and medical networks. Its detection of 32 attack types, fortified security, and intuitive interface make it a vital tool for healthcare cybersecurity. Future enhancements could include AI-driven anomaly detection and integration with hospital information systems.