**Project Title: AI-Driven Hybrid Cyber Defensive System for Intelligent Malware Detection and Threat Insight**

**Team Members • Chimdessa Tesfaye • Getachewu Getu • Meseret ghebiresilassie • Rediet Bekele**

**Machine Learning Project Documentation: Deployment**

**1. Overview**

The deployment phase represents the transition from development to production, making our malware detection model available for real-world use. For this project, we implemented an on-premises deployment solution that integrates with Windows systems to provide real-time malware detection alerts. The deployment process involved model serialization, creating a prediction service, developing a notification system, and implementing security and monitoring protocols. This allows the system to analyze Portable Executable (PE) files and immediately alert users when malware is detected, while maintaining high performance and security standards.

**2. Model Serialization**

The trained XGBoost model was serialized to enable efficient storage and deployment:

* **Serialization Format**: We used Python's ***joblib*** library (from scikit-learn ecosystem) for serialization, which is particularly efficient for storing NumPy arrays used by our model
* **File Storage**: The serialized model was saved as ***malware\_detection\_xgboost.pkl*** with a compressed format to reduce storage requirements
* **Version Control**: Each serialized model includes metadata about the training date, feature set version, and performance metrics
* **Size Considerations**: The final serialized model is approximately 126kb, making it compact enough for efficient distribution across enterprise systems
* **Dependency Management**: A requirements.txt file was created to ensure all necessary Python packages (xgboost, scikit-learn, etc.) are installed in the deployment environment

**3. Model Serving**

We implemented an ***on-premises*** solution for model serving:

* **Architecture**: The model runs as a Windows service that monitors specified directories for new PE files
* **Trigger Mechanism**: The service activates when new executables are downloaded or installed, automatically analyzing them
* **Prediction Engine**: Core components include:
  + Feature extraction module (using pefile library)
  + Loaded XGBoost model for classification
* **Resource Management**: The service is designed to use minimal system resources, with prediction times averaging under 100ms per file
* **Local Integration**: Unlike cloud solutions, this on-prem approach ensures all file analysis occurs locally without transmitting sensitive data externally

**4. API Integration**

While our primary deployment is through a Windows service, we also developed a REST API for enterprise integration:

* **Framework**: Flask-based API running locally on port 5000
* **Endpoints**:
  + /analyze (POST): Accepts PE file paths, returns malware probability
  + /batch\_analyze (POST): Processes multiple files
  + /version (GET): Returns model version info
* **Input Format**: PE apps are inserted to the model to know it is malware or not.
* **Response Format**: Whether the PE app is malware or not.
* **Example Request**:

**"file\_path": "C:/Downloads/suspicious.exe"**

 **Example Response**:

**"prediction": "malicious",**

**5. Security Considerations**

Several security measures were implemented:

* **Authentication**: API requests require HMAC-signed tokens
* **Encryption**: All API communications use TLS 1.2+ even for localhost traffic
* **File Handling**: PE files are analyzed in a sandboxed environment
* **Model Protection**: Serialized model is checksum-verified to prevent tampering
* **Data Privacy**: No file contents or features leave the local machine
* **Windows Integration**: Uses Windows Security Account Manager (SAM) for authentication in enterprise deployments

**6. Monitoring and Logging**

A comprehensive monitoring system was implemented:

* **Performance Metrics Tracked**:
  + Prediction latency (95th percentile target: <200ms)
  + CPU/Memory usage
  + Daily prediction counts
  + False positive/negative rates
* **Dashboard**:
  + Local web dashboard at <http://localhost:5001> displays real-time metrics
  + Includes ROC curve updates and feature importance visualizations
* **Model Drift Detection**:
  + Weekly checks for significant changes in feature distributions
  + Automated alerts if prediction confidence scores drop below thresholds

This deployment architecture ensures our malware detection system operates efficiently, securely, and reliably in real-world Windows environments while maintaining the high accuracy (99.88%) achieved during testing.