8-Week Intrusion Detection System (IDS) Development Roadmap

Team:

Lalith Kumar Raju Somalaraju

Shaik Abdul Gaffar

Sripathi Vamsi Krishna

Yasheswini Mallela

Week 1: Project Foundation & Data Understanding

Goal: Set up development environment and understand the CIC-IDS-2017 dataset

Tasks:

• Day 1-2: Environment setup
  - Install Python 3.8+, PyTorch, Scapy, Flask, SocketIO
  - Set up Jupyter notebooks for data exploration
  - Install Wireshark/Npcap for packet capture
  - Create project structure and version control
• Day 3-4: Dataset analysis
  - Load and explore CIC-IDS-2017 CSV files
  - Understand feature distributions and attack types
  - Identify constant/irrelevant features
  - Create data quality report
• Day 5-7: Data preprocessing pipeline
  - Implement data cleaning and normalization
  - Create train/validation/test splits
  - Build feature engineering pipeline (78 → 66 features)
  - Implement label encoding for attack types

Deliverables: Clean dataset, preprocessing pipeline, data exploration notebook

Week 2: Feature Engineering & Data Pipeline

Goal: Build robust feature extraction from network packets

Tasks:

• Day 1-3: Packet analysis framework
  - Implement packet parsing with Scapy
  - Create statistical feature extractors (flow duration, packet sizes, etc.)
  - Build timing-based features (inter-arrival times, jitter)
  - Implement protocol-specific features
• Day 4-5: Advanced feature engineering
  - Create flow-based features (bidirectional flows)
  - Implement windowing for time-series features
  - Add network topology features

- Build feature validation and alignment logic
• Day 6-7: Data pipeline optimization
  - Implement batch processing for large datasets
  - Create feature scaling and normalization
  - Build data validation and quality checks
  - Optimize memory usage for large files

Deliverables: Complete feature engineering module, data pipeline, feature validation

Week 3: Model Architecture Design

Goal: Design and implement multiple ML models for ensemble approach

Tasks:

• Day 1-2: CNN-LSTM model
  - Design architecture for sequential network data
  - Implement 1D CNN layers for local patterns
  - Add LSTM layers for temporal dependencies
  - Create attention mechanisms
• Day 3-4: Deep Neural Network (DNN)
  - Design multi-layer perceptron architecture
  - Implement batch normalization and dropout
  - Add residual connections
  - Create ensemble voting mechanisms
• Day 5-7: Autoencoder for anomaly detection
  - Design encoder-decoder architecture
  - Implement reconstruction error calculation
  - Create dynamic threshold mechanisms
  - Build anomaly scoring system

Deliverables: Model architectures, training scripts, ensemble framework

Week 4: Model Training & Optimization

Goal: Train all models with proper validation and hyperparameter tuning

Tasks:

• Day 1-3: Training infrastructure
  - Implement PyTorch Lightning modules
  - Create training loops with early stopping
  - Add model checkpointing and logging
  - Implement cross-validation
• Day 4-5: Hyperparameter optimization
  - Use Optuna or similar for hyperparameter tuning
  - Implement learning rate scheduling
  - Add data augmentation techniques

- Create model comparison metrics
• Day 6-7: Model evaluation
  - Implement comprehensive evaluation metrics
  - Create confusion matrices and ROC curves
  - Build model performance comparison
  - Implement ensemble prediction logic

Deliverables: Trained models, evaluation reports, performance metrics

Week 5: Real-time Inference Engine

Goal: Build production-ready inference system

Tasks:

• Day 1-3: Inference architecture
  - Implement model loading and caching
  - Create feature alignment and scaling
  - Build prediction pipeline
  - Add confidence scoring
• Day 4-5: Real-time processing
  - Implement packet capture integration
  - Create streaming inference pipeline
  - Add result caching and buffering
  - Build error handling and recovery
• Day 6-7: Performance optimization
  - Implement multi-threading for inference
  - Add batch processing for efficiency
  - Create memory management
  - Build monitoring and logging

Deliverables: Inference engine, real-time processing, performance benchmarks

Week 6: Live Packet Capture System

Goal: Implement robust packet capture and processing

Tasks:

• Day 1-3: Packet capture framework
  - Implement Scapy-based packet capture
  - Add interface detection and selection
  - Create packet filtering and parsing
  - Build capture statistics and monitoring
• Day 4-5: Integration with inference
  - Connect capture to inference engine

- Implement real-time feature extraction
- Add packet-to-feature mapping
- Create processing queues and buffering

• Day 6-7: Error handling and recovery
- Add capture error handling
- Implement automatic reconnection
- Create fallback mechanisms
- Build comprehensive logging

Deliverables: Live capture system, integration layer, error handling

Week 7: Web Dashboard & User Interface

Goal: Create professional, real-time dashboard

Tasks:

• Day 1-3: Backend API development
- Implement Flask web server
- Create RESTful APIs for data access
- Add SocketIO for real-time updates
- Build data serialization and caching

• Day 4-5: Frontend development
- Create responsive HTML/CSS interface
- Implement real-time data visualization
- Add interactive charts and graphs
- Create notification and alert systems

• Day 6-7: UI/UX enhancement
- Implement futuristic design theme
- Add animations and visual effects
- Create threat detection alerts
- Build offline analysis interface

Deliverables: Complete web dashboard, real-time UI, alert system

Week 8: Testing, Deployment & Documentation

Goal: Comprehensive testing, deployment, and documentation

Tasks:

• Day 1-3: Testing and validation
- Implement unit tests for all modules
- Create integration tests for full pipeline
- Add performance and stress testing
- Build test data generation tools

• Day 4-5: Deployment preparation
  - Create installation scripts and requirements
  - Build configuration management
  - Add system monitoring and health checks
  - Create backup and recovery procedures
• Day 6-7: Documentation and presentation
  - Write comprehensive user documentation
  - Create technical documentation
  - Build demo scenarios and test cases
  - Prepare presentation materials

Deliverables: Tested system, deployment package, documentation, demo

Key Milestones & Success Criteria

Week 2: Data Pipeline
  - Process CIC-IDS-2017 dataset completely
  - Extract 78 features from raw packets
  - Achieve 95%+ data quality score
Week 4: Model Performance
  - CNN-LSTM: >90% accuracy on test set
  - DNN: >85% accuracy on test set
  - Autoencoder: <5% false positive rate
Week 6: Real-time Processing
  - Process 1000+ packets/second
  - <100ms inference latency
  - 99%+ uptime during capture
Week 8: Production Ready
  - Complete system integration
  - Professional dashboard with real-time updates
  - Comprehensive testing and documentation

Technology Stack

• Backend: Python 3.8+, PyTorch, Scapy, Flask, SocketIO

• Frontend: HTML5, CSS3, JavaScript, Chart.js

• Data: CIC-IDS-2017, Pandas, NumPy

• ML: PyTorch Lightning, Optuna, Scikit-learn

• Deployment: Windows/Linux, Npcap, Wireshark

Risk Mitigation

• Week 3: Have backup model architectures ready

• Week 5: Implement fallback inference methods

• Week 6: Test on multiple network interfaces

• Week 7: Create mobile-responsive design

• Week 8: Plan for different deployment scenarios

This roadmap provides a structured approach to building a production-ready intrusion detection system with real-time capabilities, professional UI, and comprehensive testing.