Software Requirements Specification (SRS)

SmartNetIDS: A Lightweight Network Intrusion Detection System Using ML

**1. Introduction**

**1.1 Purpose**

This document outlines the software requirements for SmartNetIDS, a lightweight, intelligent, and modular Network Intrusion Detection System (NIDS). It provides an in-depth specification covering the design, development, implementation, and testing of a real-time anomaly detection system built using machine learning techniques. The system targets academic and research-based use cases and aims to demonstrate practical concepts of anomaly detection in network security.

**1.2 Scope**

SmartNetIDS captures live network traffic, processes and extracts key features, applies a machine learning (ML) model to detect anomalous patterns, and provides visualization and logging of alerts. The system prioritizes modularity, interpretability, and legal compliance. The core functionalities include: - Real-time packet sniffing - ML-based anomaly detection using Isolation Forest - Detection of specific MITRE ATT&CK techniques - Logging and alerting suspicious activities - Web-based dashboard for monitoring, visualization, and model management - Training and evaluation of models using open-source datasets (CIC-IDS 2017, NSL-KDD)

**1.3 Definitions, Acronyms, and Abbreviations**

* IDS: Intrusion Detection System
* NIDS: Network-based Intrusion Detection System
* ML: Machine Learning
* Scapy: Python library for network packet crafting/sniffing
* Streamlit: Python-based dashboard framework
* CICIDS: Canadian Institute for Cybersecurity Intrusion Detection Dataset
* NSL-KDD: Network Security Laboratory Knowledge Discovery Dataset
* PII: Personally Identifiable Information

**2. Overall Description**

**2.1 Product Perspective**

SmartNetIDS is a standalone software tool developed in Python. It uses open-source libraries to provide real-time network monitoring and anomaly detection. The architecture includes modules for packet capture, feature extraction, ML model loading and prediction, and data visualization via a web interface. The tool is not designed for enterprise use but as a learning and proof-of-concept platform for researchers and students.

**2.2 Product Features**

* **Packet Sniffing**: Continuous capture of packets using Scapy
* **Feature Extraction**: Automatic conversion of packet metadata into numeric ML features
* **ML-based Detection**: Use of Isolation Forest to identify outliers in traffic patterns
* **Alert System**: Console alerts and log file generation for suspicious packets
* **Visualization Tools**: Real-time graphs, anomaly scores, feature distributions
* **Streamlit UI**:
  + Sidebar: Detection toggle, threshold slider, manual simulator, load model
  + Home: Anomaly statistics, timeline chart, recent alerts
  + Stats: Bandwidth graphs, port histograms
  + ML Trainer: Dataset upload, retrain model, view distribution
  + Log Viewer: Log file display and keyword filtering
  + Settings: ML threshold, logging options, model load/save

**2.3 User Characteristics**

Target users include cybersecurity students, researchers, and educators with: - Knowledge of networking and basic protocols - Familiarity with Python and command-line tools - Awareness of ethical practices in cybersecurity testing

**2.4 Constraints**

* Requires administrative/root access for live packet capture
* Performance limited by host machine hardware
* May not scale effectively to large enterprise networks
* Should not be deployed on unauthorized or production networks

**2.5 Assumptions and Dependencies**

* Linux environment is assumed for packet capture support
* Public datasets are available and preprocessed
* Dependencies like Scapy, scikit-learn, Streamlit are installed

**3. Specific Requirements**

**3.1 Functional Requirements**

* FR1: The system shall capture live network packets using Scapy
* FR2: The system shall preprocess packets and extract numerical features
* FR3: The system shall load a trained ML model from disk
* FR4: The system shall classify packets as normal or anomalous using Isolation Forest
* FR5: The system shall log anomalies with timestamps and packet details
* FR6: The system shall display alerts in a web dashboard
* FR7: The system shall allow uploading a new dataset and retraining the model
* FR8: The system shall visualize bandwidth, port usage, and anomaly scores
* FR9: The system shall comply with ethical and legal standards for data collection

**3.2 Non-Functional Requirements**

* NFR1: The system shall operate with minimal resource consumption
* NFR2: The dashboard shall render updates with latency under 1 second
* NFR3: Model inference shall complete within 500ms per packet
* NFR4: All captured data must remain within the local system for privacy
* NFR5: Logging shall not contain any personally identifiable information (PII)

**3.3 Anomaly Detection Techniques**

| Anomaly Category | Detection Logic |
| --- | --- |
| Port Misuse | Match port-protocol inconsistencies |
| Packet Size | Flag extreme packet lengths |
| Protocol Violations | TCP flag misconfigurations |
| Port Scanning | Detect rapid sequential destination port access |
| DNS Tunneling | Detect abnormal DNS query rates |
| Data Exfiltration | Large outbound packet size and destination analysis |
| MITRE ATT&CK | Pattern-matching for techniques T1059, T1086, T1041, etc. |

**3.4 ML Pipeline Description**

* **Step 1**: Load and preprocess dataset (CIC-IDS2017 or NSL-KDD)
* **Step 2**: Normalize and scale features (StandardScaler)
* **Step 3**: Train Isolation Forest (or Random Forest for labeled)
* **Step 4**: Evaluate using Precision, Recall, F1-Score, Confusion Matrix
* **Step 5**: Save model with joblib
* **Step 6**: Load model in real-time packet monitor

**3.5 Ethical & Legal Requirements**

* Testing only in simulated or controlled environments
* No real-time monitoring of unauthorized or public networks
* No storage of usernames, passwords, cookies, or session data
* Full transparency of data collection and analysis methods

**3.6 Tools and Dependencies**

| Tool | Use |
| --- | --- |
| Python | Base language |
| Scapy | Packet capture and generation |
| Pandas, NumPy | Data handling and preprocessing |
| scikit-learn | ML model training and evaluation |
| Streamlit | Interactive web interface |
| Joblib | Model persistence |
| Matplotlib/Seaborn | Visualization tools |

**4. Appendices**

**4.1 Dataset Details**

* **CIC-IDS 2017**: Contains benign and attack traffic; labeled
* **NSL-KDD**: Improved KDD dataset with better class balance

**4.2 Safe Testing Practices**

| Practice | Status |
| --- | --- |
| Use of localhost/VMs | ✅ Safe |
| Use of open datasets | ✅ Safe |
| Use of public networks | ❌ Unsafe |
| Capturing third-party traffic | 🚫 Illegal |

**4.3 Compliance and Guidelines**

* OWASP Testing Guide
* SANS Institute Best Practices
* Indian IT Act 2000 (Sec 66)
* NCIIPC Guidelines

**4.4 Project Disclaimer**

“To ensure ethical compliance, all traffic used for training and testing was either synthetically generated in a virtual environment or sourced from publicly available datasets (e.g., CIC-IDS 2017). No real-time traffic from public networks or unauthorized systems was captured or analyzed. The SmartNetIDS tool was developed and tested entirely on isolated or authorized testbeds.”

**End of Document**