Contents

Contents ii

Revisions ii

1 Introduction 1

1.1 Document Purpose 1

1.2 Product Scope 2

1.3 Intended Audience and Document Overview 3

1.4 Definitions, Acronyms and Abbreviations 4

1.5 Document Conventions 4

2 Overall Description 5

2.1 Product Overview 5

2.2 Product Functionality 5

2.3 Design and Implementation Constraints 6

2.4 Assumptions and Dependencies 6

3 Specific Requirements 7

3.1 External Interface Requirements 7

3.2 Functional Requirements 8

3.3 Use Case Model 9

4 Other Non-functional Requirements 10

4.1 Performance Requirements 10

4.2 Safety and Security Requirements 10

4.3 Software Quality Attributes 10

# Introduction

This document describes the Software Requirements Specification (SRS) for the Intrusion Detection System (IDS) designed to detect zero-day attacks using Machine Learning (ML), Deep Learning (DL), and Generative AI techniques. Zero-day attacks are increasingly targeting critical sectors such as healthcare, finance, and infrastructure by exploiting unknown vulnerabilities, making them difficult to detect using traditional IDS solutions.

This SRS provides a detailed overview of the system's purpose, scope, functionality, and technical specifications. It outlines the system architecture, user interface requirements, system constraints, and the expected interaction between system components. The document is intended for developers, testers, instructors, and project stakeholders to ensure a shared understanding of system goals and functionalities.

## Document Purpose

This document outlines the software requirements for the **Intrusion Detection System for Zero-Day Attack Detection Using ML and DL Approaches**, Version 1.0. The system aims to identify unknown (zero-day) cyber threats in real-time by leveraging advanced anomaly detection and classification techniques, including Variational Autoencoders (VAE), Deep Belief Networks (DBN), Generative Adversarial Networks (GANs), and Random Forest classifiers.

The purpose of this SRS is to define the complete functionality and behavior of the IDS system. It covers the overall architecture, system components, user interactions, performance goals, constraints, and assumptions. This document serves as a reference point for developers, project members, testers, and instructors to guide the development and evaluation of the system. It focuses on the backend detection engine and data pipeline while allowing scope for future extensions such as real-time deployment and frontend integration.

## Product Scope

The proposed software is a machine learning-based **Intrusion Detection System (IDS)** specifically designed to detect **zero-day attacks**—cyber threats that exploit previously unknown vulnerabilities. The system combines unsupervised anomaly detection using **Variational Autoencoders (VAE)** and **Deep Belief Networks (DBN)** with **Generative Adversarial Networks (GANs)** for data augmentation and a **Random Forest** model for final classification. This multi-stage pipeline enables the system to learn from normal network traffic patterns, generate synthetic attack data, and classify anomalous traffic with high accuracy.

## Intended Audience and Document Overview

This Software Requirements Specification (SRS) document is intended for the following audiences:

* **Developers**: To guide them in understanding the architecture, functionalities, and required behaviors of the system during development.
* **Project Managers**: To ensure alignment of project goals with timelines and resources, helping track progress and manage expectations.
* **Testers**: To define the expected system behavior and provide clear criteria for testing the system against requirements.
* **Professors/Clients**: To evaluate the system's feasibility, scope, and the technical details of the solution in a clear, structured format.

The document is organized into several sections to provide a thorough overview of the system. It begins with an **Introduction** and **Product Scope**, followed by detailed **System Features**, **External Interface Requirements**, and **System Architecture**. The document is designed to be read sequentially, starting with the overview sections to understand the system’s goals and objectives. Developers and testers should focus on the detailed technical sections like **Functional Requirements**, **System Models**, and **Non-Functional Requirements**.

## Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| DBN | Deep Belief Network – a generative graphical model composed of multiple layers of stochastic, latent variables |
| GAN | Generative Adversarial Network – a class of machine learning frameworks for data generation |
| NSL-KDD | A benchmark dataset used for evaluating intrusion detection systems |
| RBM | Restricted Boltzmann Machine – a type of neural network used for feature learning |
| VAE | Variational Autoencoder – a type of neural network used for unsupervised learning and anomaly detection |
| Zero-Day Attack | An attack that exploits unknown or unpatched vulnerabilities in software or hardware |

## Document Conventions

## Formatting Conventions

## The primary font used in this document is Arial, with font size 11 or 12, depending on section context.

## All document text is single-spaced, with 1-inch margins on all sides.

## Section and subsection titles follow the hierarchical numbering pattern outlined in this template (e.g., 1, 1.1, 1.2, etc.).

## Italics are used for author comments, placeholders, or areas requiring user input or review.

## Bullet points and numbered lists are used to organize detailed requirements and system descriptions clearly.

## 

## Naming Conventions

## Acronyms such as IDS, VAE, GAN, and DBN are defined in Section 1.4 and used consistently throughout the document.

## Technical terms and system components follow standard terminology used in machine learning and cybersecurity domains.

## Module and subsystem names use Title Case for clarity and readability (e.g., Data Preprocessing Module, Anomaly Detection Module).

## References and Acknowledgments

This Software Requirements Specification (SRS) refers to the following documents and external resources for guidance, implementation, and evaluation:

1. **NSL-KDD Dataset Repository**  
   URL: https://www.unb.ca/cic/datasets/nsl.html  
   Description: Benchmark dataset used for training and evaluating the IDS.
2. **IEEE Std 830-1998** – IEEE Recommended Practice for Software Requirements Specifications  
   Description: Provides the standard structure and content for writing software requirements specifications.
3. **Scikit-learn Documentation**  
   URL: https://scikit-learn.org/stable/  
   Description: Used for implementing and understanding the Random Forest classifier.
4. **PyTorch Documentation**  
   URL: https://pytorch.org/docs/stable/index.html  
   Description: Deep learning framework used for building VAE, DBN, and GAN components.
5. **NIST Special Publication 800-94** – Guide to Intrusion Detection and Prevention Systems (IDPS)  
   Description: Provides a baseline understanding of IDS requirements, types, and architectures.

# Overall Description

## Product Overview

The proposed Intrusion Detection System (IDS) is a self-contained, AI-driven solution developed to enhance the detection of zero-day attacks that evade traditional signature-based IDS tools. This product is not a direct successor to any existing software but is designed as an innovative, research-based system leveraging modern deep learning and generative models. It is capable of identifying novel threats in network traffic through anomaly detection, even when those threats have never been seen before.

The system can function independently or be integrated into a larger cybersecurity framework as a modular component. It interacts with the external environment through network data capture modules, feeding processed traffic data into the internal detection pipeline. The system includes components for feature extraction (VAE-DBN), synthetic attack data generation (GAN), and attack classification (Random Forest), and outputs classification results with confidence scores. Future integration with Security Information and Event Management (SIEM) systems or alert dashboards is also possible.

## Product Functionality

The major functions that the system must perform include:

* **Anomaly Detection**: Detect deviations in network traffic using a Variational Autoencoder (VAE) to identify unknown or abnormal behavior patterns.
* **Feature Extraction**: Use a Hybrid Deep Belief Network (DBN) to extract hierarchical features from network traffic data for better anomaly detection accuracy.
* **Synthetic Attack Data Generation**: Generate realistic zero-day attack data using Generative Adversarial Networks (GANs) to improve training quality and model generalization.
* **Traffic Classification**: Classify network traffic as benign or a specific attack type (e.g., DDoS, exfiltration) using a Random Forest classifier.
* **Input Handling**: Accept network traffic input (from dataset or live traffic logs) and preprocess it for model compatibility.
* **Output Generation**: Produce labeled threat alerts with confidence scores for each detected event.
* **Self-Improvement**: Periodically retrain the models using newly generated or real-world traffic data to adapt to evolving attack patterns

## Design and Implementation Constraints

* **Hardware Limitations**: The system is designed to process high volumes of network traffic, which may require a minimum of 16 GB RAM and GPU acceleration (e.g., NVIDIA CUDA-compatible GPU) for real-time model inference and training of deep learning components.
* **Technology Stack**: The development will use Python as the primary programming language with libraries such as PyTorch (for VAE, DBN, and GAN implementation), Scikit-learn (for Random Forest classification), and Pandas/Numpy for data preprocessing. PostgreSQL or local file-based storage will be used for logging traffic and classification results.
* **Security Considerations**: The IDS must handle sensitive traffic data securely. Data input/output channels will be secured, and access to model retraining interfaces will be restricted to authorized users only.
* **Interface Requirements**: The system must interface with existing traffic data sources (e.g., PCAP files, live sniffing via Wireshark/tcpdump) and eventually integrate with a SIEM or alerting dashboard.
* **Design Methodology**: The system design must follow the COMET method (Collaborative Object Modeling and Architectural Design Method), which supports object-oriented software design by focusing on domain modeling and design patterns.  
  Reference: Hassan Gomaa, "Designing Software Product Lines with UML – From Use Cases to Pattern-Based Software Architectures", Addison-Wesley, 2004.

## Assumptions and Dependencies

* It is assumed that the project will use publicly available and maintained datasets like **NSL-KDD** or **CICIDS** for training and evaluation. Inaccessibility or poor data quality may affect model accuracy.
* We assume the availability of **GPU-enabled hardware** for training deep learning models (e.g., VAE, DBN, GAN). Without GPU access, model training and inference could become significantly slower.
* The system depends on the periodic availability of new traffic samples (real or synthetic) to retrain the models. If data sources are outdated or limited, the system’s ability to detect evolving threats may be compromised.
* We assume compatibility between the system and standard network capture tools such as Wireshark, tcpdump, or Suricata for input integration.
* The project relies on third-party open-source libraries such as PyTorch, Scikit-learn, and NumPy. Any major updates or deprecations in these libraries may require modifications to the system.

# Specific Requirements

## External Interface Requirements

### User Interfaces

The primary interface for the IDS will be a command-line or web-based dashboard used by cybersecurity professionals or system administrators. While the backend system operates autonomously, a basic user interface will allow users to interact with the system, including:

* Viewing classification results (benign, DDoS, exfiltration, etc.) with confidence scores.
* Uploading new traffic datasets (e.g., PCAP files) for evaluation.
* Triggering retraining cycles with newly available real or synthetic data.
* Viewing model performance metrics (accuracy, precision, recall) on test data.

### Hardware Interfaces

* Network Interface Cards (NICs): The system passively monitors network traffic via NICs. These interfaces are used to capture live data packets using tools like tcpdump or Wireshark.
* GPU Hardware (Optional but Recommended): Deep learning components such as VAE, DBN, and GAN benefit significantly from GPU acceleration (e.g., NVIDIA CUDA-enabled GPUs) for faster training and inference.
* Storage Devices: The system interacts with local or cloud-based storage to read/write PCAP files, logs, pre-processed datasets, and model checkpoints.
* Router or Switch (Traffic Source): While not directly controlled, the IDS depends on upstream network equipment to mirror or forward traffic for analysis.

### Software Interfaces

* **Web-Based Dashboard (Optional)**: A future-facing interface component where users can upload traffic data, view classification results, initiate retraining, and monitor system performance. The backend software exposes endpoints or APIs for interaction with this dashboard.
* **Security Information and Event Management (SIEM) Tools**: The IDS can optionally be integrated with external SIEM systems (e.g., Splunk, Elastic Stack) to send detected threats or anomalies via REST APIs or syslog.

## Functional Requirements

This section defines the specific functionalities that the Intrusion Detection System must support to detect and classify zero-day network attacks using machine learning and deep learning techniques.

* **The system shall detect anomalies in network traffic:**

The system shall use a Variational Autoencoder (VAE) model to learn patterns of normal traffic and detect deviations using reconstruction loss and KL divergence.

* **The system shall perform hierarchical feature extraction:**

The system shall use a hybrid Deep Belief Network (DBN) to pre-train the encoder of the VAE, enabling better representation learning of complex traffic patterns.

* **The system shall generate synthetic attack traffic:**

The system shall use a Generative Adversarial Network (GAN) to create synthetic zero-day attack data to augment the training dataset and improve detection robustness.

* **The system shall classify traffic as benign or malicious:**

The system shall use a trained Random Forest classifier to categorize network traffic into benign or specific attack classes (e.g., DDoS, data exfiltration).

* **The system shall produce a confidence score with every classification:**

Each classification result shall be accompanied by a confidence score indicating the certainty level of the model's prediction*.*

## Use Case Model

### Use Case #1 (use case name and unique identifier – e.g. U1)

**Author –** Chitransh Mathur

**Purpose** - To identify deviations from normal network traffic using a Variational Autoencoder (VAE) model supported by a Deep Belief Network (DBN) for feature learning. The goal is to detect potential zero-day attacks in real-time.

**Requirements Traceability –**F1, F2, F5

**Priority** - High — this is a core function of the IDS and must operate reliably for the system to be effective.

**Preconditions**

* The VAE-DBN model must be trained on normal network traffic.
* Pre-processed and formatted network traffic must be available for analysis.

**Post conditions** –

* The system flags traffic as normal or anomalous.
* Detected anomalies are forwarded for classification or alert generation.

**Actors** –

* Network Traffic Monitor (system-level actor)
* Security Analyst (human actor)
* VAE-DBN Engine (internal system actor)

**Extends –** None

**Flow of Events**

* 1. Basic Flow –
* Network traffic is continuously collected.
* Data is preprocessed (cleaned, normalized).
* The VAE-DBN model analyzes the traffic.
* Reconstruction error and KL divergence are calculated.
* An anomaly is flagged if the error exceeds a threshold.
  1. Alternative Flow –
* If traffic data is missing or incomplete, skip analysis and log an error.
* If the system is retraining, queue incoming traffic for later analysis.
  1. **Includes** (other use case IDs)
* U2 – Classify Traffic using Random Forest  
  U3 – Alert Generation with Confidence Score
* Threshold for anomaly detection needs fine-tuning based on evaluation metrics

# Other Non-functional Requirements

## Performance Requirements

## Safety and Security Requirements

S1. The system shall ensure secure storage and access of network traffic data, including logs and model outputs, by encrypting all data at rest using AES-256 encryption.

S2. All communications between system components (e.g., data collection agents and analysis module) shall be protected using TLS 1.3 to prevent data interception or tampering during transmission.

S3. The system shall require authentication of users through secure login credentials and role-based access control (RBAC). Only authorized users (e.g., system admins or security analysts) shall be able to view, modify, or initiate model retraining.

## Software Quality Attributes

The system shall be modular in design, allowing each core component (e.g., VAE engine, GAN module, classifier) to be updated or replaced independently.  
To support this:

* Each component will follow a plug-and-play architecture, with well-documented APIs for communication between modules.
* Code will be written following PEP8 and modular design principles for readability and long-term maintainability.
* Configuration files (e.g., YAML or JSON) will allow retraining thresholds, dataset paths, and model parameters to be updated without changing the core codebase.