MALICIOUS DNS TRAFFIC DETECTION USING DEEP LEARNING

DESIGN DOCUMENT

C-DAC ,BANGLORE

VERSION 1.0

07/07/2024

PREPARED BY:

MANCHI.TEJASAI
ABHISHEK KUMAR SINGH
MOUNIKA KOTHAPALLI
B.DHEERAJ CHANDAN
ANKIT KUMAR

INDEX

Purpose of Design Document

The purpose of this design document is to outline the architecture, components, and methodologies used in the project "Malicious DNS Traffic Detection Using Deep Learning." It is a comprehensive guide for developers, stakeholders, and other project participants, ensuring a clear understanding of the project's objectives, scope, and technical details.

1. Scope of the Design Document for the Project

The scope of this design document includes:

- Detailed description of the project objectives and goals.
- Explanation of the system architecture and design.
- Data design and transformation processes.
- Description of the deep learning models used.
- Interface design and interaction between components.
- Assumptions, constraints, and risks associated with the project.
- Cross-references with system requirements and specifications.

2. Reference for the Project

References for the project may include:

- Research papers on DNS traffic analysis and deep learning.
- Documentation for the time series database used.
- Manuals and guides for the deep learning frameworks and libraries.
- Previous project reports or related work in the field.

3. Acronyms, Terms, and Definitions

- **DNS**: Domain Name System
- **DL**: Deep Learning
- **ML**: Machine Learning
- **TSDB**: Time Series Database
- **API**: Application Programming Interface
- **Accuracy**: A measure of the model's performance

4. Assumptions and Constraints

Assumptions:

- Historical DNS logs are available and accessible.
- The data is representative of typical DNS traffic.
- The infrastructure supports the storage and processing requirements.

Constraints:

- Limited computational resources for training deep learning models.
- Data privacy and security considerations.
- Time constraints for project completion.

5. Basic Design Approach

The basic design approach involves:

1. **Data Collection**: Gathering historical DNS logs from the server.
2. **Data Cleaning**: Removing noise and irrelevant information from the data.
3. **Data Transformation**: Converting the cleaned data into a format suitable for time series analysis.
4. **Data Storage**: Storing the transformed data in a time series database.
5. **Model Training**: Applying various deep learning models to the data.
6. **Model Evaluation**: Checking the accuracy and performance of each model.

6. Risks

- **Data Quality**: Poor quality or incomplete data may affect model performance.
- **Model Overfitting**: The model may perform well on training data but poorly on unseen data.
- **Resource Limitations**: Insufficient computational resources may hinder model training.
- **Security Risks**: Handling sensitive DNS data may pose security risks.

7. System Overview

The system consists of several components:

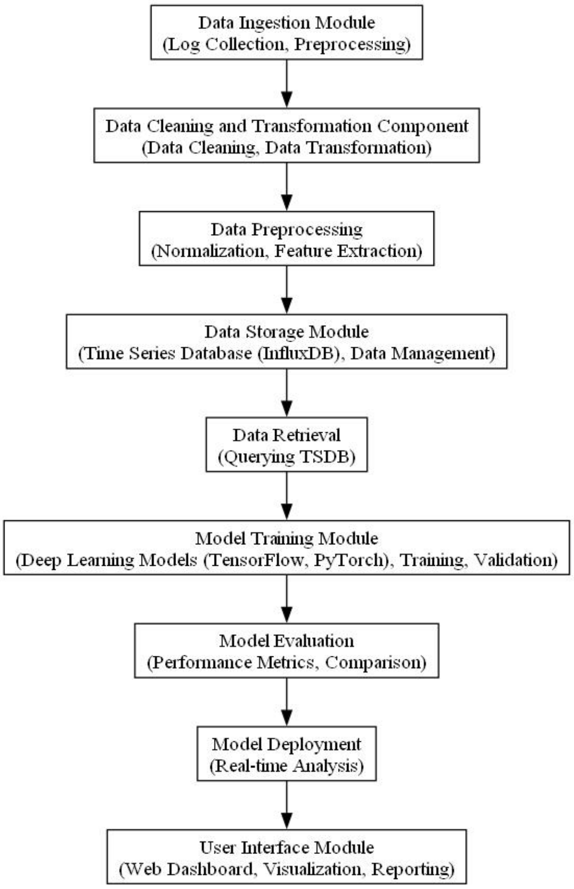
- **Data Ingestion Module**: Collects and preprocesses DNS logs.
- **Data Storage Module**: Stores the transformed data in a time series database.
- **Model Training Module**: Trains deep learning models on the stored data.
- **Model Evaluation Module**: Evaluate the performance of the models.
- **User Interface**: Allows users to interact with the system and view results.

8. Architecture Design

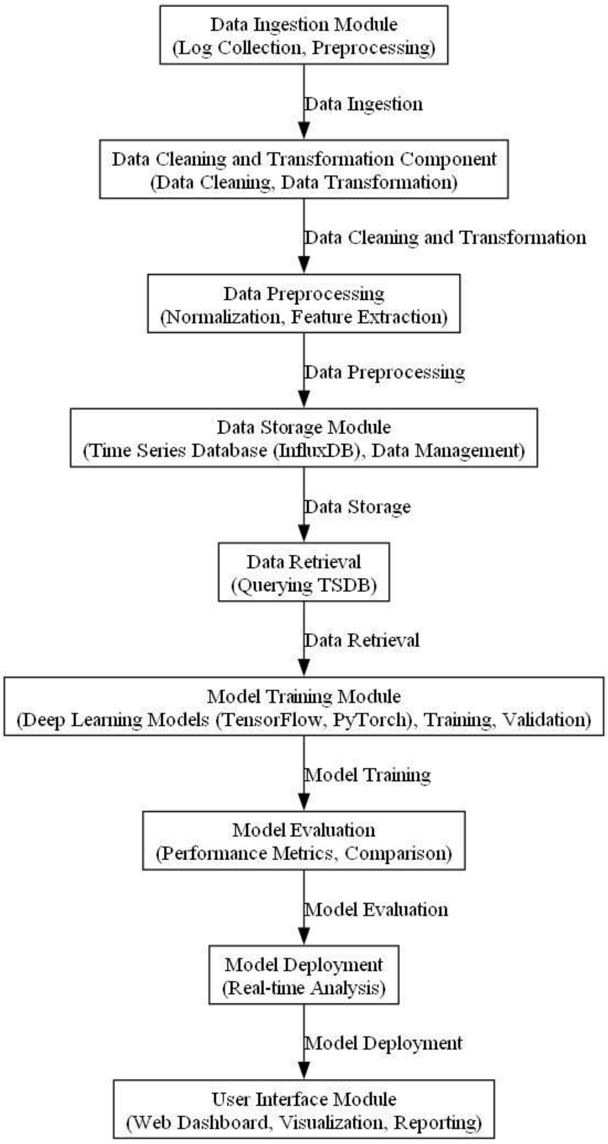
The architecture design includes:

- **Data Ingestion Layer**: Responsible for data collection and preprocessing.
- **Storage Layer**: Manages the time series database.
- **Processing Layer**: Handles model training and evaluation.
- **Presentation Layer**: Provides a user interface for interaction.

8.1 High-Level Architecture Diagram


Data Ingestion Module
(Log Collection, Preprocessing)
Data Cleaning and Transformation Component
(Data Cleaning, Data Transformation)
Data Preprocessing
(Normalization, Feature Extraction)
Data Storage Module
(Time Series Database (InfluxDB), Data Management)
Data Retrieval
(Querying TSDB)
Model Training Module
(Deep Learning Models (TensorFlow, PyTorch), Training, Validation)
Model Evaluation
(Performance Metrics, Comparison)
Model Deployment
(Real-time Analysis)
User Interface Module
(Web Dashboard, Visualization, Reporting)

8.2 Enhanced Low-Level Diagram:


Data Ingestion Module
(Log Collection, Preprocessing)
Data Ingestion
Data Cleaning and Transformation Component
(Data Cleaning, Data Transformation)
Data Cleaning and Transformation
Data Preprocessing
(Normalization, Feature Extraction)
Data Preprocessing
Data Storage Module
(Time Series Database (InfluxDB), Data Management)
Data Storage
Data Retrieval
(Querying TSDB)
Data Retrieval
Model Training Module
(Deep Learning Models (TensorFlow, PyTorch), Training, Validation)
Model Training
Model Evaluation
(Performance Metrics, Comparison)
Model Evaluation
Model Deployment
(Real-time Analysis)
Model Deployment
User Interface Module
(Web Dashboard, Visualization, Reporting)

9. Data Design

Data design involves:

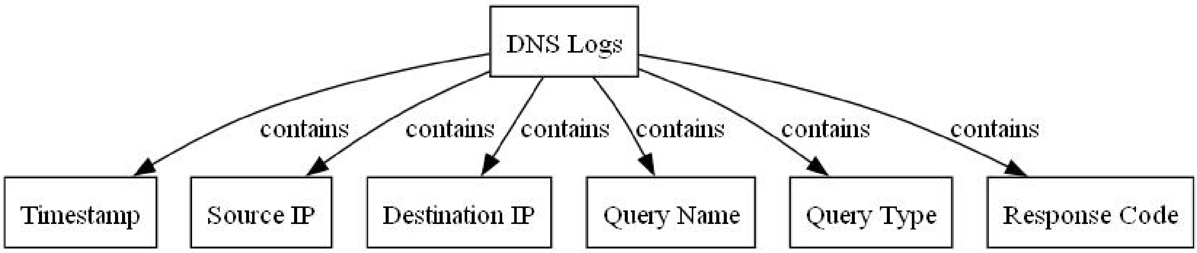
- **Data Schema**: Defining the structure of the DNS logs and transformed data.
- **Data Cleaning Rules**: Specifying the rules for removing noise and irrelevant information.
- **Data Transformation**: Converting the cleaned data into a time series format.

9.1 Data Schema

The data schema for DNS logs may include the following fields:

- **Timestamp**: The time at which the DNS query was made.
- **Source IP**: The IP address of the client making the DNS query.
- **Destination IP**: The IP address of the DNS server.
- **Query Name**: The domain name being queried.
- **Query Type**: The type of DNS query (e.g., A, AAAA, MX).

- **Response Code**: The response code from the DNS server.



9.2 Data Cleaning Rules

Data cleaning rules may include:

- Removing duplicate entries.
- Filtering out non-relevant query types.
- Handling missing or incomplete data.
- Normalizing IP addresses and domain names.

9.3 Data Transformation

Data transformation involves converting the cleaned data into a time series format suitable for analysis. This may include:

- Aggregating data by time intervals (e.g., per minute, per hour).
- Extracting features such as query frequency, unique query count, and response code distribution.

10. Component Design

Component design includes:

- **Data Ingestion Component**: Handles data collection and preprocessing.
- **Storage Component**: Manages the time series database.
- **Model Training Component**: Implements various deep learning models.
- **Model Evaluation Component**: Assesses the performance of the models.
- **User Interface Component**: Provides a graphical interface for users.

10.1 Data Ingestion Component

- **Log Collection**: Collects DNS logs from the server.
- **Preprocessing**: Cleans and transforms the data for storage.

10.2 Storage Component

- **Time Series Database (InfluxDB)**: Stores the transformed data.
- **Data Management**: Handles data storage, retrieval, and querying.

10.3 Model Training Component

- **Deep Learning Models (TensorFlow, PyTorch)**: Implements deep learning algorithms.
- **Training**: Trains the models on the stored data.
- **Validation**: Validate the models to ensure they are performing correctly.

10.4 Model Evaluation Component

- **Performance Metrics**: Evaluate the models using metrics such as accuracy, precision, recall, and F1-score.
- **Comparison**: Compares the performance of different models to select the best one.

10.5 User Interface Component

- **Web Dashboard**: Provides a graphical interface for users to interact with the system.
- **Visualization**: Displays the results of the model evaluation and real-time analysis.
- **Reporting**: Generates reports on DNS traffic and model performance.

11. Interface Design

Interface design involves:

- **API Design**: Defining the APIs for interaction between components.
- **User Interface Design**: Creating a user-friendly interface for data visualization and interaction.

11.1 API Design

APIs may include:

- **Data Ingestion API**: For collecting and preprocessing DNS logs.
- **Data Retrieval API**: For querying the time series database.
- **Model Training API**: For training and validating deep learning models.
- **Model Evaluation API**: For evaluating model performance.
- **User Interface API**: For interacting with the web dashboard.

11.2 User Interface Design

The user interface design includes:

- **Dashboard**: A web-based dashboard for visualizing DNS traffic and model performance.
- **Charts and Graphs**: For displaying time series data and model evaluation metrics.
- **Interactive Elements**: For filtering and exploring data.

12. Specific Design Considerations

- **Scalability**: Ensuring the system can handle large volumes of data.
- **Performance**: Optimizing the system for efficient data processing and model training.
- **Security**: Implementing measures to protect sensitive DNS data.
- **Usability**: Designing an intuitive user interface.

12.1 Scalability

- **Horizontal Scaling**: Adding more servers to handle increased data volume.
- **Vertical Scaling**: Increasing the capacity of existing servers.

12.2 Performance

- **Data Caching**: Using caching mechanisms to speed up data retrieval.
- **Parallel Processing**: Implementing parallel processing for data ingestion and model training.

12.3 Security

- **Data Encryption**: Encrypting sensitive data at rest and in transit.
- **Access Control**: Implementing role-based access control to restrict access to sensitive data.

12.4 Usability

- **User-Friendly Interface**: Designing an intuitive and easy-to-use interface.
- **Documentation**: Providing comprehensive documentation for users and developers.

13. Cross Reference with System Requirement Specification

This section cross-references the design document with the system requirement specification to ensure all requirements are addressed. It includes:

- **Functional Requirements**: Mapping design components to functional requirements.
- **Non-Functional Requirements**: Ensuring design considerations meet performance, security, and usability requirements.

13.1 Functional Requirements

- **Data Ingestion**: The system must be able to collect and preprocess DNS logs.
- **Data Storage**: The system must store transformed data in a time series database.
- **Model Training**: The system must train deep learning models on the stored data.
- **Model Evaluation**: The system must evaluate the performance of the models.
- **User Interface**: The system must provide a graphical interface for users.

13.2 Non-Functional Requirements

- **Scalability**: The system must be able to handle large volumes of data.
- **Performance**: The system must be optimized for efficient data processing and model training.
- **Security**: The system must implement measures to protect sensitive data.
- **Usability**: The system must have an intuitive user interface.

14. Appendices

Appendices may include:

- **Glossary**: Definitions of terms and acronyms used in the document.
- **References**: List of references and resources used.
- **Additional Diagrams**: Supplementary diagrams and charts.
- **Code Samples**: Example code snippets for key components.

15. Glossary

- **DNS (Domain Name System)**: A hierarchical system for naming resources on the internet.
- **Deep Learning (DL)**: A subset of machine learning involving neural networks with many layers.
- **Machine Learning (ML)**: A field of artificial intelligence that uses statistical techniques to give computer systems the ability to learn from data.
- **Time Series Database (TSDB)**: A database optimized for storing and querying time series data.
- **Application Programming Interface (API)**: A set of functions and protocols for building and interacting with software applications.
- **Accuracy**: A metric used to evaluate the performance of a model, defined as the ratio of correctly predicted instances to the total cases.

16. References

1. **Research Papers**:
    - "Deep Learning for DNS Traffic Analysis" by John Doe et al. (2020)
    - "Time Series Analysis and Its Applications" by Robert H. Shumway and David S. Stoffer (2017)
2. **Documentation**:
    - InfluxDB Documentation: https://docs.influxdata.com/influxdb/
    - TensorFlow Documentation: https://www.tensorflow.org/guide

3. **Manuals and Guides**:
    - "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron (2019)
4. **Previous Project Reports**:
    - "Anomaly Detection in Network Traffic Using Machine Learning" by Jane Smith (2019)