REVA UNIVERSITY
Bengaluru, India

School of Computing Science and Engineering

Course Code: B20EF0701	Major Project Progress Report – SEE	Academic Year: 2023-24
		Semester: 7th

Project Group Members Details

Group No:	Group Name:

Sl. No	SRN	Full Name	Sec	Mobile No.	Email_ID	Sign
1	R20EF402	CHANDAN M S	H	6363224034	chandanms1312@gmail.com	
2	R20EF414	MOHAMMAD FAWAZ BEEJADY	H	6366926710	mohammadfawazbeejady@gmail.com	
3	R20EF436	SUMAN S	H	9380899173	sgrsuman90@gmail.com	
4	R20EF424	ROSHAN B S	H	6361594870	r20201179.roshanbs@cse.reva.edu.in	

Task Distribution

Project Leader	CHANDAN M S
Documenter Lead:	MOHAMMAD FAWAZ BEEJADY
Development Lead:	SUMAN S ROSHAN B S

Project Details

Project Title:	**Kafka-based Intrusion Detection System System**
Project Type:	KAFKA-BASED

Guide Details:	**Name: SOUJANYA B K** **Designation: Assistant Professor**	**Mobile No:9448728458**
Place of Project Work:	REVA UNIVERSITY	
External Guide:	**Name:** **Designation:**	**Mobile No:**
Remarks by Guide:		**Guide Signature** **Date:**

Contents

1. Project Planning

- Install Apache Kafka and set up a Kafka cluster. This will act as the messaging middleware.
- Create Kafka topics that will hold the streaming data.
- Write a Kafka producer script in Python that will push data to the Kafka topics in real time. This data can come from sensors, applications, databases etc.
- Write a Kafka consumer script in Python that will subscribe to the Kafka topics and read the streaming data.
- Build your anomaly detection model in Python. This could be something like Isolation Forest, One Class SVM, Local Outlier Factor etc. Train the model on historical data.
- In the Kafka consumer script, feed the streaming data to the anomaly detection model to detect anomalies in real time.
- Whenever an anomaly is detected, the consumer script can take appropriate actions like sending alerts, notifications, logging the anomalies etc.
- The whole system will work in real time as data comes from the producer and is analyzed by the consumer using the anomaly detection model.
- Kafka's distributed and fault tolerant architecture ensures high throughput and availability of the system.
- The streaming data pipeline is thus set up to detect anomalies in real time using Apache Kafka for streaming and Python for modeling.

2. Project Scope

General Project Information

Project Name: Kafka-based Intrusion Detection System

Problem/Opportunity Statement:

The organization generates a large amount of data from various sources like sensors, applications, databases etc. There is a need to detect anomalies and abnormalities in this data in real time to take corrective actions and avoid losses.

Business Benefits:

- Reduce losses due to undetected anomalies and abnormalities.
- Improve efficiency of processes by taking corrective actions as soon as anomalies occur.
- Gain insights from anomaly patterns to optimize systems and processes.
- Increase visibility into the health and performance of various systems and applications.
- Improve customer experience by reducing the impact of anomalies.

The key is to build a real time anomaly detection system that can detect issues as soon as they happen, provide alerts and notifications, maintain logs, visualize anomalies, and help take corrective actions - ultimately improving efficiency, reducing losses and gaining insights.

3. Project Deliverables

1. Anomaly Detection Model

This is the core of the system. The anomaly detection model built in Python using an appropriate algorithm like Isolation Forest, One Class SVM, Local Outlier Factor etc. The model is trained on historical data to learn what constitutes 'normal' behavior and anything different is flagged as an anomaly.

The model is tuned for high accuracy in detecting anomalies while keeping false alarms to a minimum. It can detect both point anomalies and contextual anomalies.

2. Kafka Infrastructure

The Apache Kafka cluster setup which acts as the messaging middleware. It allows data to stream in real time from producers to the anomaly detection model acting as a consumer.

Topics are created to categories different data streams. The Kafka cluster is scalable and fault tolerant to ensure high throughput and availability of the system.

3. Alerts and Notifications

The mechanism to notify relevant people when anomalies are detected. This could be emails, SMS, push notifications, slack messages etc. It helps draw attention to issues as soon as they occur.

The details of the alerts can include information like anomaly score, type of anomaly, data values, timestamps etc.

4. Anomaly Logs and Dashboards

Logs are maintained containing details of all anomalies detected - timestamps, scores, possible reasons, actions taken etc. This helps in analysis and troubleshooting.

Visual dashboards are created to show graphs, charts, and tables with live anomaly data for easy monitoring. They provide a high-level view of the system's performance.

4. Methodology

Overall Approach:

The objective is to make real-time predictions with incoming stream data from Apache Kafka using python.

We Generate representative data for training and testing the anomaly detection model. To do so, we utilize an outlier generator to create simulated transaction data. We include two variables representing the characteristics of each transaction. For this step, we develop an automated data generator to simulate real-time transactions. After that, we incorporate auto-incremental IDs, machine learning model input features, and timestamps for each transaction.

Machine Learning Model Training:

The purpose of this is to train an Isolation Forest model for anomaly detection. We Implement the Isolation Forest algorithm from the scikit-learn library. We then Use the generated data to train the model. We develop the train.py script to train the Isolation Forest model. We save the result in a .joblib file.

Apache Kafka Configuration:

We Set up Apache Kafka to facilitate real-time data streaming. We Use Kafka topics to organize and manage data streams. We Configure Kafka with appropriate parameters for the study. We create two Kafka topics, one for transactions and another for anomalies. We then Configure Kafka partitions and replication factors for optimal performance.

Anomaly Detector:

We Design a consumer to process transactions, make predictions, and filter anomalies. We develop an anomaly detector script for Kafka consumption and anomaly detection. We then Utilize multiprocessing for parallel processing. We Read messages from the transactions topic and Use the Isolation Forest model to predict anomalies. We then send outlier transactions to the anomalies topic.

Slack Notification:

We Implement a mechanism to notify anomalies through Slack. We Utilize the Slack API for message notifications. We then create a consumer to read from the anomalies topic and send alerts. To accomplish this, we develop an alerts script to read anomalies and send Slack notifications. We then Integrate with the Slack API using the provided token.

5. Modules identified

Data Generation Module:

This module focuses on creating simulated transaction data for both training the anomaly detection model and testing its real-time capabilities. The data generation module includes an outlier generator to mimic diverse transaction scenarios. Key variables, such as transaction characteristics and timestamps, are incorporated to enhance the realism of the generated data.

Machine Learning Model Training Module:

The machine learning model training module centers around implementing the Isolation Forest algorithm for anomaly detection. Leveraging the scikit-learn library, this module includes the development of a training script (train.py) to train the Isolation Forest model using the generated data. The trained model is saved for subsequent real-time use.

Apache Kafka Configuration Module:

This essential module involves configuring Apache Kafka to serve as the backbone for real-time data streaming. Topics, including "transactions" and "anomalies," are created to organize and manage data streams. Configuration parameters for partitions and replication factors are set to optimize Kafka's performance.

Data Producer Module:

The data producer module is responsible for sending transaction data to the Kafka "transactions" topic. Implemented using the confluent-Kafka Python package, the data producer introduces a probability mechanism for generating outliers within the transaction data. The producer script (producer.py) simulates real-time data streaming, emphasizing the generation of outlier transactions.

Anomaly Detector Module:

The anomaly detector module encompasses the development of a consumer script (anomalies_detector.py) designed to process transactions, make predictions using the trained Isolation Forest model, and filter anomalies. Utilizing multiprocessing for parallel processing, this module reads messages from the "transactions" topic, predicts anomalies, and sends outlier transactions to the "anomalies" topic.

Slack Notification Module:

The Slack notification module facilitates the communication of detected anomalies. The consumer script (bot_alerts.py) within this module reads messages from the "anomalies" topic and utilizes the Slack API for sending notifications. It integrates with the Slack channel, providing real-time alerts for identified anomalies.

Experimental Setup

Training Module:

The training module involves utilizing a representative dataset to train and evaluate the Isolation Forest model. The module focuses on key methods, such as training the model using the generated data and assessing its performance based on predefined metrics.

Real-time Testing Module

This module assesses the real-time performance of the anomaly detection system. It includes methods for streaming simulated transactions to evaluate system responsiveness and monitoring the accuracy of anomaly predictions in real-time.

6. Requirement and cost Analysis

Requirements Analysis:

1. **Functional Requirements:**
   - Identify the key functional requirements mentioned in the blog. This could include aspects such as data streaming, anomaly detection algorithms, and real-time processing capabilities.
2. **Non-functional Requirements:**
   - Look for non-functional requirements like performance, scalability, and reliability. Consider factors such as the volume of data, processing speed, and system availability.
3. **Technology Stack:**
   - List the technologies used in the implementation. This may include Apache Kafka, Python libraries for machine learning, and any other relevant tools mentioned in the blog.
4. **Integration Points:**
   - Identify any external systems or components that need to be integrated into the solution. This could include data sources, databases, or other services.
5. **Security Requirements:**
   - Check if the blog provides information on security measures implemented in the system, such as data encryption, access controls, and authentication.

Cost Analysis:

1. **Hardware and Infrastructure:**
   - Summarize the hardware requirements mentioned in the blog. This could include server specifications, storage, and network requirements.
2. **Software Licensing:**
   - Check if there are any costs associated with software licenses for tools or libraries used in the solution. For example, some machine learning libraries may have licensing fees.
3. **Development and Implementation:**
   - Estimate the development effort required based on the complexity of the solution. Consider factors such as coding, testing, and deployment.
4. **Training and Documentation:**
   - If the blog mentions any training or documentation costs, include them in your analysis. Training may be required for the development team or end-users.
5. **Maintenance and Support:**
   - Estimate ongoing maintenance and support costs. This could include server maintenance, bug fixes, and updates to the system.
6. **Scaling Costs:**
   - Consider costs associated with scaling the system, especially if the volume of data is expected to increase over time. This may include additional hardware or cloud service costs.
7. **Contingency and Unforeseen Costs:**
   - Factor in a contingency for unforeseen circumstances or additional costs that may arise during the implementation and maintenance phases.

7. Developing project execution timelines – Schedule

Project Execution Timelines

Project Scope:

Define the scope of the project, outlining the key objectives, deliverables, and milestones. This sets the foundation for developing the project execution timelines.

Project Work Breakdown Structure (WBS):

Create a detailed Work Breakdown Structure that decomposes the project into smaller, manageable tasks. Each task should be clearly defined and associated with specific deliverables.

Task Dependencies:

Identify dependencies between tasks. Some tasks may be dependent on the completion of others. Understanding these dependencies is crucial for realistic timeline planning.

Resource Allocation:

Allocate resources (human, technological, and financial) to each task. Consider the availability and skills of team members, as well as the tools and technologies required for the project.

Timeline Development:

1. **Task Duration:**
    - Estimate the time required for each task based on historical data, expert judgment, or any specific information provided in the blog.
2. **Gantt Chart:**
    - Create a Gantt chart illustrating the project timeline. This visual representation will help in understanding the sequence of tasks and their durations.

Critical Path Analysis:

Identify the critical path, which represents the sequence of tasks that must be completed on time for the overall project to stay on schedule. Tasks on the critical path have zero slack or float.

Milestones:

Define key milestones in the project, such as the completion of major tasks, testing phases, and deployment. Milestones provide a way to track progress and ensure that the project is on schedule.

Contingency Planning:

Factor in contingency time for unforeseen delays or issues that may arise during the project. This ensures that the project timeline remains realistic and achievable.

Review and Approval Process:

Establish a process for reviewing and approving the project execution timeline. This may involve collaboration with stakeholders, team leads, and project sponsors.

8. Conclusion

In conclusion, a Kafka-based intrusion detection system gives a more robust and secure approach in countering intrusion attacks. Kafka's distributed and fault-tolerant mechanisms can efficiently collect, process, and analyze large volumes of login requests and data. Additionally, the system's ability to handle high-throughput data streams ensures timely identification of potential and on-going attacks resulting in an overall secured system. As organizations continue to face increasingly sophisticated cyber threats, adopting a Kafka-based intrusion detection system is a good foundation to meet the evolving needs of cybersecurity.

9. References

1. T. H. Hai and N. T. Khiem, "Architecture for IDS Log Processing using Spark Streaming*" 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE),* Istanbul, Turkey, 2020, pp. 1-5, doi: 10.1109/ICECCE49384.2020.9179188.

2. Elmasry, Wisam &Akbulut, Akhan& Zaim, "A Design of an Integrated Cloud-based Intrusion Detection System with Third Party Cloud Service" 2021 Open Computer Science. 11. 365-379. 10.1515/comp-2020-0214.

3. B. Debnath et al, "LogLens: A Real-Time Log Analysis System," 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, 2018, pp. 1052-1062, doi: 10.1109/ICDCS.2018.00105.

4. L. Wirz, R. Tanthanathewin, A. Ketphet and S. Fugkeaw, "Design and Development of A Cloud-Based IDS using Apache Kafka and Spark Streaming" 2022 19th International Joint Conference on Computer Science and Software Engineering (JCSSE), Bangkok, Thailand, 2022, pp. 1-6, doi: 10.1109/JCSSE54890.2022.9836264.

5. Chi Wai Chan, "Building a Cloud-Native Feed Streaming System with Apache Kafka and Spark on Alibaba Cloud - Part B: Streaming Processing" https://www.alibabacloud.com/blog/building-a-cloud-native-feed-streaming-system-with-apache-kafka-and-spark-on-alibaba-cloud-part-b-streaming-processing_596925 November 23, 2020.

6. Prasad Alle, "Real-time Stream Processing Using Apache Spark Streaming and Apache Kafka on AWS." https://aws.amazon.com/blogs/big-data/real-time-stream-processing-using-apache-spark-streaming-and-apache-kafka-on-aws/ 30 September, 2016

7. L. Chen, M. Xian, J. Liu and H. Wang, "Intrusion Detection System in Cloud Computing Environment," 2020 International Conference on Computer Communication and Network Security (CCNS), Xi'an, China, 2020, pp. 131-135, doi: 10.1109/CCNS50731.2020.00037.

8. Y. Mehmood, M. A. Shibli, U. Habiba and R. Masood, "Intrusion Detection System in Cloud Computing: Challenges and opportunities," 2013 2nd National Conference on Information Assurance (NCIA), Rawalpindi, Pakistan, 2013, pp. 59-66, doi: 10.1109/NCIA.2013.6725325.

9. S. Roschke, F. Cheng and C. Meinel, "Intrusion Detection in the Cloud," 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, Chengdu, China, 2009, pp. 729-734, doi: 10.1109/DASC.2009.94.

10. D. Kassimi et al., "A New Approach Based on a Multi–Agent System for IDS in Cloud Computing," 2022 Ninth International Conference on Software Defined Systems (SDS), Paris, France, 2022, pp. 1-8, doi: 10.1109/SDS57574.2022.10062930.