

Anomaly Detection in Cloud Servers

Team 4 SLACKERS
Unnati Aggarwal (002741568)
Shantanu Sachdeva (002748942)

WHY?

The purpose of this project is to leverage telecom network logs to identify and flag anomalous activities, ultimately enhancing cybersecurity threat detection. By analyzing these logs, the project aims to develop algorithms and models that can automatically detect patterns indicative of potential threats or abnormalities within the network. This proactive approach can help network administrators and security teams to quickly identify and respond to potential cybersecurity incidents, thereby strengthening the overall security posture of the telecom network.

Use Case

- Network Administrator inputs stream of telecom server logs to the pub/sub topic (Updated: Apache Kafka Topic) to detect network attacks.
- Server Health Check software runs a batch of input logs hourly to detect a spike in outliers and emails the generated report to the administrator. (Updated: We plan to send daily updates of the network profile)

Data Sources

- We used the KDD Cup 1999 Dataset
- Raw TCP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN.
- Data is peppered with multiple attacks

Attacks fall into four main categories with 24 further subtypes:

- DOS: denial-of-service, e.g. syn flood; (Teardrop, Back, Land, Smurf, Neptune, Pod)
- R2L: unauthorized access from a remote machine, e.g. guessing password;
(guess_passwd, map, spy, phd, warezmaster, wazerclient, multi hop, ftp_write)
- U2R: unauthorized access to local superuser (root) privileges, e.g., various "buffer overflow" attacks; (perl, rootkit, buffer_overflow, load module)
- Probing: surveillance and other probing, e.g., port scanning. (Nmap, portsweep, ipsweep, satan)

Features

<i>feature name</i>	<i>description</i>	<i>type</i>
duration	length (number of seconds) of the connection	continuous
protocol type	type of the protocol, e.g. tcp, udp, etc.	discrete
service	network service on the destination, e.g., http, telnet, etc.	discrete
src bytes	number of data bytes from source to destination	continuous
dst bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong fragment	number of ``wrong'' fragments	continuous
urgent	number of urgent packets	continuous
<i>feature name</i>	<i>description</i>	<i>type</i>
hot	number of ``hot'' indicators	continuous
num failed logins	number of failed login attempts	continuous
logged in	1 if successfully logged in; 0 otherwise	discrete
num compromised	number of ``compromised'' conditions	continuous
root shell	1 if root shell is obtained; 0 otherwise	discrete
su attempted	1 if ``su root'' command attempted; 0 otherwise	discrete
num root	number of ``root'' accesses	continuous
num file creations	number of file creation operations	continuous
num shells	number of shell prompts	continuous
num access files	number of operations on access control files	continuous
num outbound cmd	number of outbound commands in an ftp session	continuous
is hot login	1 if the login belongs to the ``hot'' list; 0 otherwise	discrete
is guest login	1 if the login is a ``guest'' login; 0 otherwise	discrete
<i>feature name</i>	<i>description</i>	<i>type</i>
count	number of connections to the same host as the current connection in the past two hours	continuous
<i>Note: The following features refer to these same-host connections.</i>		
serror rate	% of connections that have ``SYN'' errors	continuous
rerror rate	% of connections that have ``REJ'' errors	continuous
same srv rate	% of connections to the same service	continuous
diff srv rate	% of connections to different services	continuous
srv count	number of connections to the same service as the current connection in the past two hours	continuous
<i>Note: The following features refer to these same-service connections.</i>		
srv serror rate	% of connections that have ``SYN'' errors	continuous
srv rerror rate	% of connections that have ``REJ'' errors	continuous
srv diff host rate	% of connections to different hosts	continuous

Data Sources

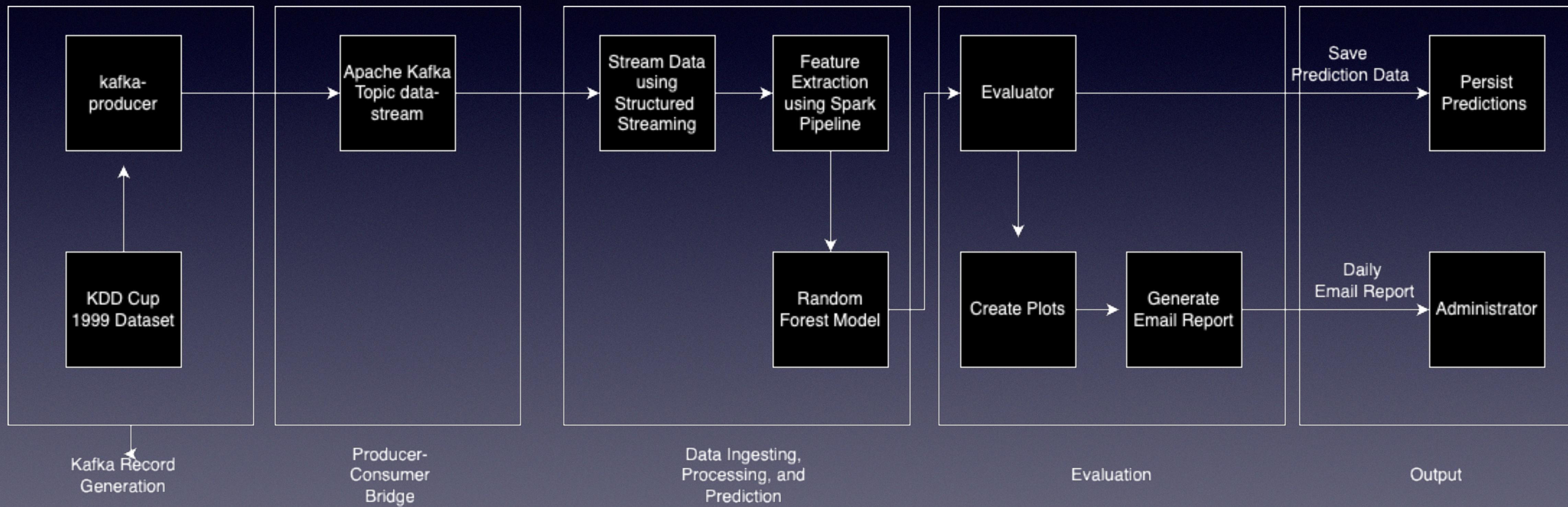
- Initially, we plan to use a pipeline of simulated data using DataFlow.
- Eventually, we aim to use live data streams.
- Update: We decided not to use simulated data, as simulated data may introduce bias into the model, as it is not diverse enough and does not cover all relevant scenarios. This can lead to overfitting, where the model performs well on the training data but poorly on new, unseen data. This would have resulted in a model that fails to perform well in real-world applications.

Methodology

Machine learning-based anomaly detection solution by highlighting the following key components:

1. Generating synthetic data to simulate production volume using Dataflow and Pub/Sub. (Updated: We implemented the Pub/Sub infrastructure using Google Cloud, but had to adopt Apache Kafka instead, due to high costs of cloud infrastructure, lack of documentation and connectors used with Google Cloud Pub/Sub)
2. Update: We introduced structured streaming using Apache Spark. A read stream context is always alive, and ready to process new logs from the Apache Kafka topic.
3. Extracting features and real time prediction using Scala/Spark.
4. Training and normalizing data using BigQuery ML's built-in k-means clustering model.(Updated: Due to high cost of using BigQuery, we switched to Apache Spark Machine Learning Libraries. We trained and tested Logistic Regression and Random Forest model)
5. Update: To send the daily network log report, we set up a DNS name and SPF record, we first created a DNS A record pointing to our server's IP address for the domain, and then added a TXT record containing the SPF policy to authorize our mail server to send emails on behalf of our domain.

Updated Methodology Diagram



Milestones

- 21st March: Establishing log schemas and mocking data to the Pub/Sub in real time.
Establishing the feature extraction criteria on aggregated logs using Scala/Spark.
(Updated: Implemented Apache Kafka Topic, data publisher and subscriber programs)
- 28th March: Creating the K-Means model using BQ ML. **(Updated: We implemented Logistic Regression and Random Forest Model. We switched to Random Forest Model as our data was labelled)**
- 4th April : Testing with outlier mock date to see if the model is able to detect it
- 11th April : Setting up output email notification
- 18th April : Final Presentation

The Big Question, Scala

- Data Publishing
- Data Ingestion
- Feature Extraction

Acceptance Criteria

- Data loss during ingestion from pub/sub topic should be less than 10% (Satisfied for our use case, but unsure if it can scale)
- Ensure the prediction model has a minimum accuracy of 90% in identifying outliers (Not Satisfied: We are able to achieve an accuracy of 71.3% using random forest on test data)

Goals

- Getting hands-on and understanding streaming analytics.
- Working with Google managed services like Dataflow, BigQuery ML
- Writing efficient and Scalable code in Scala
- Getting domain knowledge of networks, and anomaly detection using networks logs

Test Cases for Random Forest Model

The screenshot shows a Scala IDE interface with the following details:

- Title Bar:** csye7200-project - RandomForestClassifierTest.scala [spark-streaming]
- Project View:** Shows the project structure with files like LogisticRegression.scala, RandomForestClassifier.scala, and RandomForestClassifierTest.scala.
- Code Editor:** Displays code for RandomForestClassifierSpec, including imports from org.scalatest and org.apache.spark.sql, and a call to .setNumTrees(10).
- Run Tab:** Set to RandomForestClassifierSpec.
- Test Results:** A tree view showing 8 tests passed, each with a green checkmark and a duration (e.g., 50 sec 276 ms).
 - RandomForestClassifierSpec
 - getEventDataSchema
 - should return a StructType with specific schema
 - readData
 - should read data from CSV file and return DataFrame
 - getStringColumns
 - getNumericalColumns
 - createProcessingPipeline
 - should create and save pipeline model
 - createRandomForestClassifier
 - fitAndSaveModel
 - should fit the model and save it
 - predictAndEvaluate
 - should predict and evaluate the model
- Output Console:** Shows log output from the test run, including INFO messages from ShuffleBlockFetcherIterator, Executor, TaskSetManager, TaskSchedulerImpl, DAGScheduler, and SparkContext, along with accuracy and error rate metrics.
- Bottom Navigation:** Includes tabs for Git, Run, Python Packages, TODO, Problems, Terminal, Services, Profiler, Build, and sbt shell.
- Status Bar:** Shows "Tests passed: 8 (6 minutes ago)" and system information like 3688:23 LF UTF-8 2 spaces.

Test Cases for Logistic Regression Model

The screenshot shows a Java IDE interface with the following details:

- Title Bar:** csye7200-project – LogisticRegressionTest.scala [spark-streaming]
- Project Explorer:** Shows the project structure with files like RandomForestClassifier.scala, apache-flink-datasource..., LogisticRegressionTest.scala, kafka-producer..., and KafkaDataCSVTest.scala.
- Run Tab:** LogisticRegressionTestSpec
- Test Results:** A tree view showing 8 tests passed, each with a green checkmark and a duration. The tests include:
 - getEventDataSchema
 - readData
 - getStringColumns
 - getNumericalColumns
 - createProcessingPipeline
 - createLogisticRegression
 - fitAndSaveModel
 - predictAndEvaluate
- Output Console:** Displays log messages from the test execution, including INFO logs from TaskSchedulerImpl, DAGScheduler, and SparkContext, along with accuracy and error rate metrics.

```
24/04/17 16:52:20 INFO TaskSchedulerImpl: Removed TaskSet 194.0, whose tasks have all completed successfully
24/04/17 16:52:20 INFO DAGScheduler: ResultStage 194 (collectAsMap at MulticlassMetrics.scala:40) finished in 1.34 s
24/04/17 16:52:20 INFO DAGScheduler: Job 184 is finished. Cancelling potential speculative executors
24/04/17 16:52:20 INFO TaskSchedulerImpl: Killing all running tasks in stage 194: Stage finished
24/04/17 16:52:20 INFO DAGScheduler: Job 184 finished: collectAsMap at MulticlassMetrics.scala:40
--- Logistic Regression ---
Accuracy Rate = 0.6802
Error Rate = 0.3198
0.680180979418027

24/04/17 16:52:21 INFO SparkContext: Invoking stop() from shutdown hook
24/04/17 16:52:21 INFO SparkContext: SparkContext is stopping with exitCode 0.
24/04/17 16:52:21 INFO SparkUI: Stopped Spark web UI at http://10.110.244.229:4040
24/04/17 16:52:21 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
24/04/17 16:52:21 INFO MemoryStore: MemoryStore cleared
24/04/17 16:52:21 INFO BlockManager: BlockManager stopped
24/04/17 16:52:21 INFO BlockManagerMaster: BlockManagerMaster stopped
24/04/17 16:52:21 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
24/04/17 16:52:21 INFO SparkContext: Successfully stopped SparkContext
24/04/17 16:52:21 INFO ShutdownHookManager: Shutdown hook called
24/04/17 16:52:21 INFO ShutdownHookManager: Deleting directory /private/var/folders/cp/vx4f0
```
- Bottom Status Bar:** Shows "Tests passed: 8 (4 minutes ago)" and other system information like time and file path.

Test Cases for Kafka Producer

The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** csye7200-project – KafkaDataCSVTest.scala [kafka-producer]
- Project Tree:** Shows a project named "kafka-producer" with a "pub-sub" module containing Python files: createSubscription.py, createTopic.py, FakerPublisher.py, Publisher.py, and Subscriber.py.
- Editor:** The code editor displays the Scala file KafkaDataCSVTest.scala, specifically the "generateEventData" test method. The code is annotated with green arrows indicating execution flow.
- Run Tab:** The "KafkaDataCSVTest" tab is selected in the run configuration dropdown.
- Run Results:** The results pane shows the test output:
 - Tests passed: 6 of 6 tests – 32 ms
 - Testing started at 5:06 PM ...
 - Details of individual test cases:
 - should return Some(EventData) for valid input: 32 ms
 - should return None for invalid input: 32 ms
 - should return None for empty input: 32 ms
 - should return None for out-of-range value: 23 ms
 - should return Some(EventData) for zero duration: 6 ms
 - should return correct EventData for valid input: 1ms
- Status Bar:** At the bottom, it says "Process finished with exit code 0".
- Bottom Navigation:** Standard PyCharm navigation bar with Git, Run, Services, Profiler, Build, sbt shell, and other tools.

Test Cases for CreatePlot

The screenshot shows an IDE interface with the following details:

- Title Bar:** csye7200-project – CreatePlotSpec.scala [spark-streaming]
- Project Explorer:** Shows a project structure with folders like commits, offsets, sources, .metadatacrc, and metadata, along with a plots folder containing Evaluator.scala, ProcessingLogs.scala, CreatePlotSpec.scala (selected), LogisticRegressionTest.scala, logFile1.csv, CreatePlot.scala, and pieChart.png.
- Code Editor:** Displays the content of CreatePlotSpec.scala, which includes imports and a class definition for CreatePlotSpec extending AnyFlatSpec with Matchers.
- Run Tab:** Shows a green checkmark indicating "Tests passed: 2 of 2 tests – 20 sec 193ms".
- Test Results:** A tree view showing the test hierarchy and execution times:
 - Root: CreatePlotSpec (20 sec 193 ms)
 - Child: CreatePlot (20 sec 193 ms)
 - Child: should create a pie chart for attack distribution (19 sec 718 ms)
 - Child: CreatePlot (19 sec 718 ms)
 - Child: CreatePlot (475 ms)
 - Child: should create a bar chart for distribution of signature of attacks (475 ms)
- Output Panel:** Displays the command used to run the tests and the log output from Spark's default log4j profile. The log output includes information about the Spark context, Java version, and various INFO messages related to resource management and security.
- Bottom Navigation:** Includes tabs for Git, Run, Python Packages, TODO, Problems, Terminal, Services, Profiler, Build, sbt shell, and a status bar showing the current time (12:1), file encoding (LF), character set (UTF-8), and code spaces (2 spaces).

Test Cases for Utils

The screenshot shows a Scala IDE interface with the following details:

- Title Bar:** csye7200-project – UtilsSpec.scala [spark-streaming]
- Project Explorer:** Shows the project structure under 'spark-streaming' with files like Evaluator.scala, ProcessingLogs.scala, CreatePlotSpec.scala, UtilsSpec.scala, LogisticRegressionTest.scala, logFile1.csv, and CreatePlot.sc.
- Code Editor:** Displays the content of UtilsSpec.scala, which imports ... and defines a class UtilsSpec extending AnyFlatSpec with Matchers.
- Run Tab:** Set to 'UtilsSpec'.
- Output Panel:** Shows the test results:
 - Tests passed: 1 of 1 test – 34ms
 - Testing started at 6:20 PM ...
 - Moved test.csv to src/test/resources/storeCSV
 - Process finished with exit code 0
- Bottom Navigation:** Includes tabs for Git, Run, Python Packages, TODO, Problems, Terminal, Services, Profiler, Build, sbt shell, and a status bar indicating Tests passed: 1 (moments ago).