UNIVERSITY OF WATERLOO

Detecting Malware from IoT Devices

Technical Report

Course: Big Data Management Systems and Tools

Group 9 Members:
Patrick Rieux
Filsan Musa
Ashraf Ali Shaik
Wouemgny(Ron) Tagne Kamga
Yuting Gao
Andrea Quintana

Supervised By:
Karen Khalsa
Melissa Singh

Table of Contents

Objective

To stream the IoT device data to Kafka and then to data storage platform for analyzing and detecting malware.

Setting up a Publish-Subscribe Model

Selecting Tools

To set up a Publish-Subscribe Model, we first need to decide on tools to stream our data for analyzing. There are a few tools available to us. We have mainly reviewed the following tools.

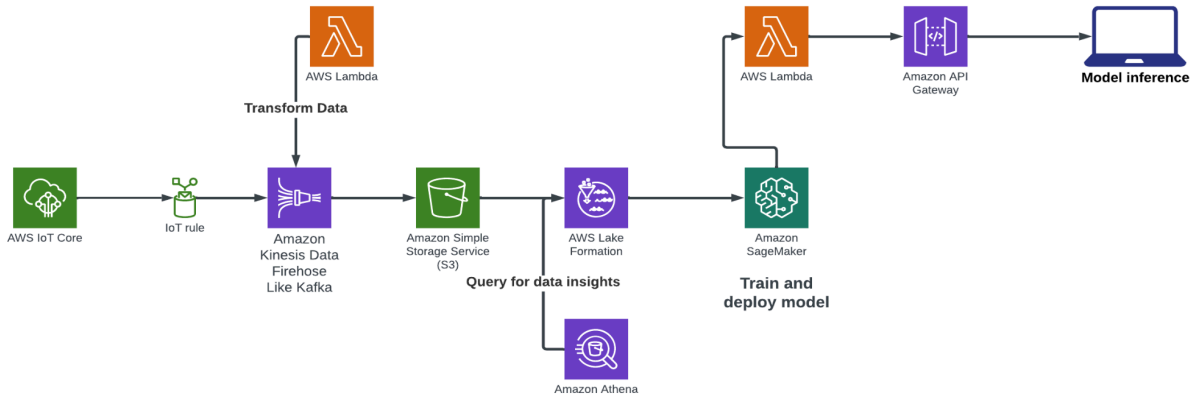
AWS

- Key benefit: Easy to set-up
- Fee: Requires a paid account

Analysis: At the beginning of our project, we considered utilizing AWS, but based on the project's time, scope, and understanding it requires a paid account, we shifted our attention to other providers. However, we found the architecture and benefits of AWS services remain significant and are worth detailed exploration. This streamlined guide below provides a comprehensive overview of the project's architecture and the benefits of AWS services.

Project Architecture

The architecture of this project leverages various AWS services to handle data ingestion, processing, storage, querying, machine learning, and API integration.



IoT Devices

- **Internet of Things (IoT)**:
  - Network of interconnected devices that collect and exchange data, including sensors, actuators, and other smart devices.
  - AWS IoT User Guide
- **Role in the Project**:
  - **Data Generation**: Primary source of data. Devices generate logs, metrics, and other telemetry data for malware detection.
  - **Data Transmission**: Ensuring secure and reliable transmission to the cloud for further processing and analysis.
- **Challenges**:

    - **Security**: Vulnerable to attacks; ensuring data integrity is critical.
    - **Scalability**: Requires scalable cloud infrastructure.
    - **Interoperability**: Different communication protocols across devices.

Data Ingestion with Kinesis Firehose

- **Amazon Kinesis Data Firehose**:
    - Fully managed service for real-time data ingestion, capturing, transforming, and loading streaming data.
    - Amazon Data Firehose
- **Role in the Project**:
    - **Data Stream Ingestion**: Ingests data from IoT devices.
    - **Transformation and Delivery**: Transforms data in-flight before delivery to AWS Lake Formation.
- **Benefits**:
    - **Real-Time Processing**: Ingests and processes data in near real-time.
    - **Scalability**: Automatically handles varying data volumes.

Data Processing with AWS Lambda

- **AWS Lambda**:
    - Serverless compute service that runs code in response to events, managing the underlying compute resources automatically.
    - AWS Lambda Developer Guide
- **Role in the Project**:
    - **Event-Driven Processing**: Triggers functions by data from Kinesis Firehose to parse, filter, and apply initial malware detection logic.
    - **Flexibility and Scalability**: Automatically scales to handle data volumes.
- **Benefits**:
    - **Automatic Scaling**: Handles any data throughput level.
    - **Cost-Effective**: Pay only for the compute time used.
    - **Integration**: Easily integrates with other AWS services.

Data Storage with AWS Lake Formation

- **AWS Lake Formation**:
    - Service for setting up, securing, and managing data lakes.
    - AWS Lake Formation Developer Guide
- **Role in the Project**:
    - **Data Organization**: Stores processed data securely for further analysis.
    - **Data Security**: Fine-grained access control and encryption.
- **Features**:
    - **Data Cataloging**: Automatically catalogs data for easy discovery and governance.
    - **Access Control**: Detailed permissions management.
- **Benefits**:
    - **Simplified Management**: Streamlines data lake setup and management.
    - **Enhanced Security**: Ensures secure data storage and access.
    - **Scalability**: Supports large-scale data storage and management.

Querying Data with AWS Athena

- **Amazon Athena**:
  - Interactive query service to analyze data in Amazon S3 using SQL.
  - Amazon Athena Developer Guide
- **Role in the Project**:
  - **Data Analysis**: Enables SQL queries to analyze stored data and identify malware patterns.
  - **Schema Detection**: Integrates with AWS Glue Data Catalog for schema and data type detection.
- **Configuration Steps**:
  - **Creating an S3 Bucket**: Store data to be queried.
  - **Setting Up a Database in Athena**: Organize datasets.
  - **Creating Tables in Athena**: Define tables mapping to S3 data.
  - **Running Queries**: Use SQL for data analysis.
- **Best Practices**:
  - **Partition Data**: Improves query performance.
  - **Use Columnar Formats**: Optimizes performance and reduces costs.
  - **Update Statistics**: Helps Athena optimize query execution.

Model Creation with AWS SageMaker

- **Amazon SageMaker**:
  - Fully managed service for building, training, and deploying machine learning models.
  - Amazon SageMaker Developer Guide
- **Role in the Project**:
  - **Model Training**: Trains models with data from AWS Lake Formation.
  - **Model Deployment**: Deploys models to SageMaker endpoints for real-time inference.
- **Features**:
  - **Integrated Jupyter Notebooks**: Collaborative environment for data exploration and model building.
  - **Built-in Algorithms**: Pre-built algorithms for large-scale datasets.
  - **Automated Model Tuning**: Adjusts hyperparameters for better performance.
  - **One-Click Deployment**: Simplifies model deployment.
- **Benefits**:
  - **Streamlined Workflow**: Integrates all steps of machine learning.
  - **Scalability**: Handles large datasets and complex models.
  - **Cost Efficiency**: Pay only for resources used.
  - **Flexibility**: Supports custom algorithms and frameworks.

Exposing Model via API Gateway and Lambda

- **Amazon API Gateway**:
  - Fully managed service for creating, publishing, maintaining, monitoring, and securing APIs.
  - Amazon API Gateway Developer Guide
- **AWS Lambda**:

    - Serverless compute service that runs code in response to API requests.
- **Role in the Project**:
    - **API Creation**: API Gateway creates an endpoint for external applications to interact with machine learning models.
    - **Model Querying**: Lambda functions query SageMaker model endpoint and return inference results.
- **Benefits**:
    - **Easy Integration**: Simplifies integration with external applications.
    - **Scalability**: Automatically handles varying traffic levels.
    - **Security**: Provides mechanisms for securing APIs.

Advantages of AWS Native Serverless Systems

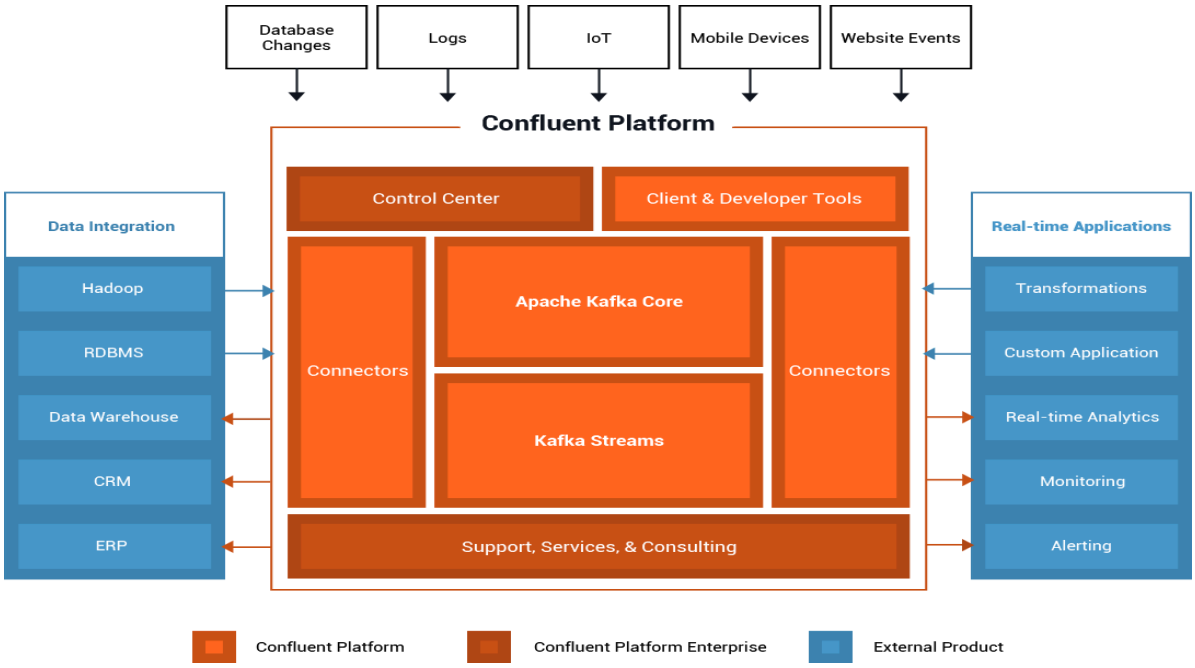
- **Cost Efficiency**:
    - **Pay-as-You-Go**: Reduces costs by paying only for compute time and resources used.
    - **No Infrastructure Management**: Further reduces operational costs by eliminating server management.
- **Scalability**:
    - **Automatic Scaling**: Ensures optimal performance during peak times without manual intervention.
- **Flexibility and Agility**:
    - **Rapid Deployment**: Enables quick development and deployment of applications.
    - **Microservices Architecture**: Allows for manageable, independent components.
- **Security**:
    - **Built-in Security Features**: Ensures secure data handling and regulatory compliance.

Confluent and DataBricks

- Key benefit: Easy to administrate.
- Fee:
    - Confluent: Free tier is available ($400 at signups to spend during the first 30 days).
    - Databricks: Offers Community Edition, a free version of the Databricks Platform that does not require cloud resources.

Analysis: After ruling out AWS, we seeked for free alternatives that can help us accomplish what we are looking for, then Confluent and Databricks came into the view.

Confluent is a commercial version of an online Kafka service and offers fully managed cloud-native data streaming, which fulfills the project requirements. It also offers Kafka support, admin portal, productivity tools, plugins and modules. Compared to Apache Kafka, it is serverless and can retain data at any scale without growing computing. It automatically rebalances partitions for better performance, and has role-based access control and audit logs to improve security and detect threats.



Databricks' Community Edition was introduced and practiced in our course, so all group members have a higher level of familiarity about how to operate on the platform.

Decision

Confluent and Databricks together offer real-time stream processing and transformation, which is built on Kafka, and provide a workspace for analyzing. After considering the benefits of both technologies, we decided to proceed with these tools.

Setting up the Model

Architecture

Source: IoT device data

Event Streaming and Processing: Confluent

Analytics: Databricks

In Confluent, we first created a new cloud environment and set up the cluster, topic and client in the environment. To push our data to Confluent, we created a Kafka Producer. A Kafka Producer is a client application that publishes or writes events to a Kafka cluster. For our project, we used one of the group member's computers as publisher and created a Jupyter Notebook to push data to Confluent. Each group member also created a user specific API in Confluent. In Databricks, we were then able to connect to the Confluent cloud using Bootstraperserver, Topic, APIKey and APISecret information.

Challenge - Governance

Different from traditional Kafka, Confluent offers more flexibility in terms of governance. These additional features were convenient but created some challenges for new users. Therefore, when setting up the cluster in Confluent, to allow all users being able to create user specific API, we reviewed and compared the governance setups in both Kafka and Confluent. The below is a summary of our findings. For this project, we assigned every group member with ResourceKeyAdmin and Organization Admin roles to allow a smooth API creation, other amendments and administration in the cluster.

<u>Kafka</u>

- **Authorizations roles:**
  - Guest: Metrics Viewer, Data Discovery
  - Developer: ResourceKeyAdmin, Organization Admin, Link Developer
  - Administration: AccountAdmin, BillingAdmin
  - IT support: Operator, NetworkAdmin

<u>Confluent</u>

- **Authorizations roles:**
  - Guest: Metrics Viewer, Data Discovery
  - Developer: ResourceKeyAdmin, Organization Admin, Link Developer
  - Administration: AccountAdmin, BillingAdmin
  - IT support: Operator, NetworkAdmin
- **Data quality:**
  - Schema - Define and enforce universal data standards that enable scalable data compatibility. Schema versioning for change management of ingress and egress data
  - Data validation - Enforce rules, report rejections to ensure ongoing management of failures/changes.
  - Share schemas to facilitate implementation of new producers and subscribers
- **Stream Catalog:**
  - Metadata tagging allows searching and securing messages
  - Discovery portal facilitates data exploration with API that allows searching for entities
- **Stream mapping**
  - Graphical maps for visual understanding, exploring and analysis
  - Historical logs for troubleshooting and security event forensics

Data Analysis

Exploratory Data Analysis

The data used in the analysis was found on Kaggle. The Malware_IoT_log_file provided us with 7 CSV files for detecting malware.

These CSV files include a number of attributes but were stored as one value in Confluent. So, in Databricks, we first parsed the string and created a clean table for analysis. The clean table contains the 26 attributes. After the clean table was developed, we noticed there were duplicate records and removed these records. We also reviewed the columns with missing values. Column index','duration_hour','duration_minute','duration_second','total_duration_milliseconds','orig_bytes', and 'resp_bytes' showed an over 50% missing value. As a significant number of records were missing in these columns, which would influence our Model if maintained, we removed these columns. Column resp_pkts and resp_pkts also had some missing values but were less significant, so we replaced the missing values with medians.

We then dropped columns 'ts','ts_date', 'ts_time','duration','duration_day','duration_milisecond', 'key', and 'uid'. These columns didn't provide us with valuable information to identify if a malware is malicious or not. Unique values were also checked for. We noticed there were 420 unique records in 'id_orig_h', 3148 unique records in 'id_resp_h', and 42 unique records in 'history'. Since these are all data type objects, we decided to remove these columns for better model learning. Upon examining column 'service', 'local_resp', 'tunnel', we figured that there were over 80% unknowns in these columns and determined dropping these columns would improve our model performance. To understand the correlation between the attributes, we created the correlation matrix displayed below in Figure 1. The analysis helped us identify the most influential features in the dataset and understand their relationships, which was crucial for optimizing the performance of the model we looked to develop.

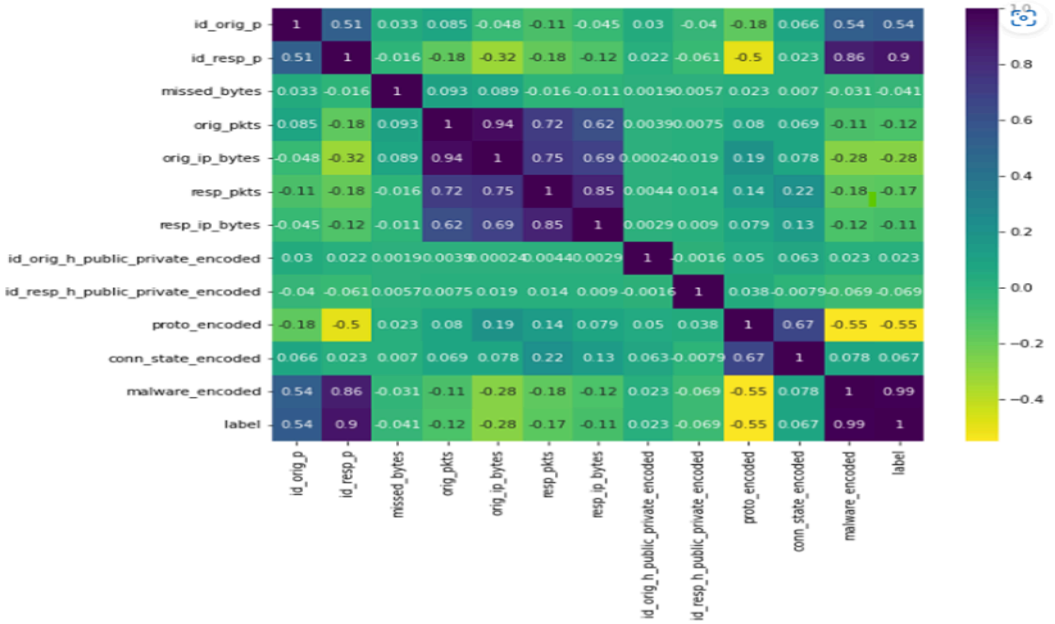


Figure 1. Heatmap

Key observations from the Figure 1 analysis include:

- High Correlation Features: Certain features, such as id_resp_p and label, as well as orig_ip_bytes and orig_pkts, show strong positive correlations. These features would be critical in predicting whether a connection is malicious or benign, as they have significant relationships with the target variable.
- Negative Correlations: The feature proto_encoded exhibits a moderate negative correlation with malware_encoded, suggesting that certain protocol types may be inversely related to the likelihood of a connection being classified as malware.

Decision Trees

Grid Search

In our analysis, we utilized GridSearchCV to fine-tune the hyperparameters of the Decision Tree model. GridSearchCV systematically tested every possible combination of specified hyperparameters, including 'max_depth', 'criterion', and 'splitter', to identify the optimal configuration for maximizing model accuracy. This method was effective because it ensured a comprehensive search, guaranteeing that the best possible parameters were found. The results from this process showed perfect performance metrics, with 100% accuracy, precision, recall, and F1-score. The interpretation of these results suggests that the model was highly effective in predicting malware with the given data. However, the perfect scores indicate that the model might have overfit the training data, and further validation would be necessary to ensure it generalizes well to new, unseen data.

Random Search

Following the GridSearchCV, we employed RandomizedSearchCV to further explore hyperparameter optimization. Unlike GridSearchCV, RandomizedSearchCV selects random combinations of hyperparameters within a specified range, allowing for a broader and faster search. This method is particularly effective when the parameter space is large, as it can identify good hyperparameter settings with reduced computational cost. The RandomizedSearchCV yielded results similar to GridSearchCV, with 100% accuracy, precision, recall, and F1-score. These perfect results once again indicate that the model was highly successful in classifying the data, but they also raise concerns about potential overfitting, which could impact the model's performance on new datasets. Further evaluation would be necessary to confirm the model's robustness.

Both GridSearchCV and RandomizedSearchCV effectively optimized our Decision Tree model, achieving perfect performance metrics. GridSearchCV offered a comprehensive search across all hyperparameters, ensuring the best possible configuration but at a higher computational cost. In contrast, RandomizedSearchCV provides a quicker, more efficient search by sampling random parameter combinations, making it ideal for larger parameter spaces or when resources are limited. While both methods were successful, RandomizedSearchCV stands out for its efficiency, making it the more practical choice when balancing performance with computational constraints.

Ensemble Models

Random Forest Classifier

In addition to the previous methods, we implemented a Random Forest Classifier as part of our ensemble modeling approach. The Random Forest algorithm builds multiple decision trees during

training and combines their results to improve the overall accuracy and robustness of the model. This approach is particularly effective in reducing overfitting, as it averages out the predictions from several trees, making the model more generalizable to unseen data.

The results obtained from the Random Forest Classifier were exceptional, with 100% accuracy, precision, recall, and F1-score, as shown in the attached screenshot. These results indicate that the Random Forest Classifier was highly successful in correctly classifying all instances in the test set, suggesting a strong model performance. However, the perfect metrics also warrant consideration of potential overfitting, similar to the other models tested, and further validation on different datasets would be advisable to confirm its robustness.

Gradient Boosted Trees

Following the Random Forest approach, we explored a different ensemble method using Gradient Boosted Trees. Gradient Boosting is a powerful technique that builds the model in a sequential manner, where each new tree attempts to correct the errors made by the previous ones. This iterative process allows for a more refined model that can capture complex patterns in the data.
The results from the Gradient Boosted Trees were also outstanding, with perfect scores of 100% in accuracy, precision, recall, and F1-score, as seen in the attached screenshot. The perfect results indicate that Gradient Boosted Trees were highly effective in classifying the data accurately. However, similar to previous models, the perfect metrics raise concerns about potential overfitting, and further testing on different datasets would be necessary to confirm the model's generalizability. Both Random Forest Classifier and Gradient Boosted Trees delivered perfect results, but each has its strengths. The Random Forest is robust and reduces overfitting by averaging multiple trees, making it ideal for diverse data. Gradient Boosted Trees, with its sequential learning, captures complex patterns more effectively, offering a refined model for intricate data relationships. While Gradient Boosting may excel in nuanced modeling, Random Forest’s simplicity and generalization make it a strong choice when efficiency is key. Further validation is needed to confirm their generalizability in different scenarios.