Redis Enterprise and AWS Fraud Detection Solution Overview

redis aws

Solution Brief

Introduction

Detecting fraud is a very complex process. For any given transaction or activity, the system needs to decide whether it's fraudulent or not and take action within seconds. Failing to do so can lead to significant losses. Fraudsters are evolving everyday and are moving in tandem with digital banking transformations, discovering innovative ways to steal or fake customers' identities, and commit fraud. In this piece, among many challenges, we'll focus on the **two** main ones – false positives and latency. Both yield unhappy customers and substantial loss to sellers.

1. False Positives

A false positive is when the system flags a legitimate transaction by the user as fraud. This is very frustrating for the customer and very costly for the seller.

Effective machine learning algorithms, such as random cut forest and xgboost, are very effective in combating fraud. However, as fraud constantly evolves, for better detection, along with continual model learning, a multi-layer approach needs to be in place.

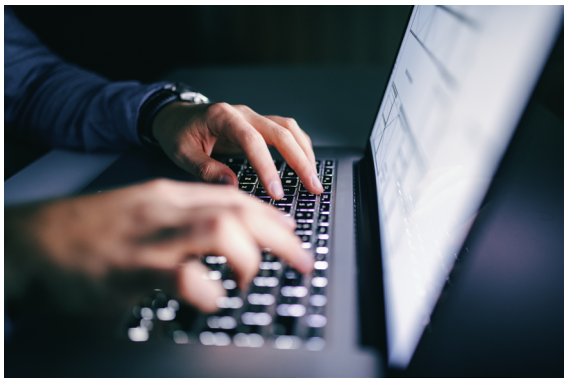
2. Latency

As mentioned above, fraudsters are evolving and becoming more complex. The detection should not fall behind and enhanced detection should not increase latency.

If companies cannot detect whether the transaction is fraudulent or not within a few seconds, by default, it's considered genuine. Therefore latency is very important in fraud detection.

Multi-Layer Approach

While the ML models need to be regularly updated with the help of automated MLOps pipelines, detecting fraud can be enhanced with a multi-layered rule-based approach.



Below are some examples of these rules:

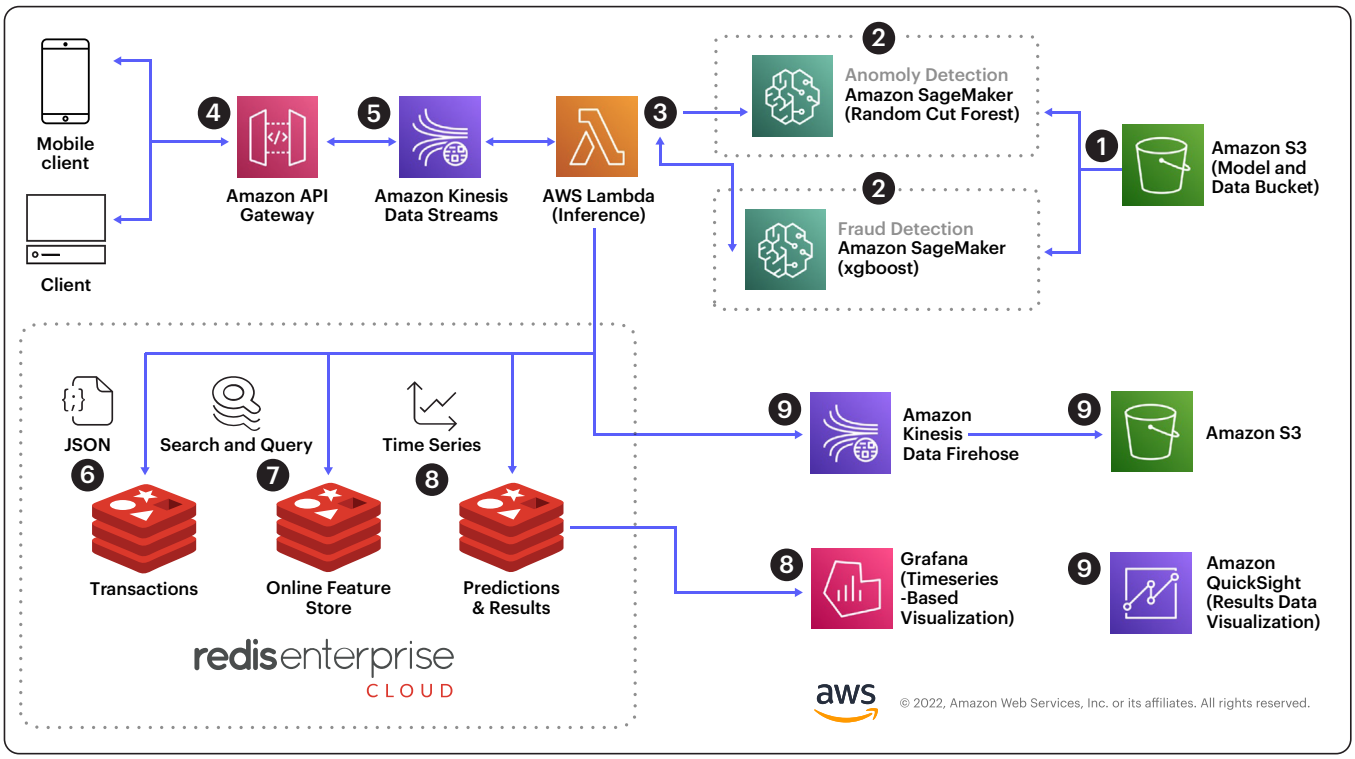
1. Blacklisting fraudsters' IP addresses
2. Deriving and utilizing the latitude and longitude data from users' IP addresses
3. Utilizing the data on browser type and version, as well as operating system, active plugins, timezone and language
4. *Per user purchase profile:* Has this user made purchases in these categories before?
5. *General purchase profiles*: Has this type of user made purchases in these categories before?

The rules can be implemented in a way that it can start from a "low cost" to "high cost". If a user makes a purchase within already made categories and within min/max amounts, the app can tag the transaction as non-fraudulent and eliminate the cycles to be spent on further rules and the ML layer.

Alternatively, we can simultaneously run the transaction through the rule-based and ML layers and combine the outcomes. In terms of computation, this approach will be more costly but as it runs concurrently, the latency will not be increased.

Solution Architecture

Below depicts the solution implemented in Amazon's AWS cloud.



1. An Amazon Simple Storage Service (Amazon S3) bucket contains historical datasets of credit card transactions.
2. An Amazon SageMaker notebook instance with different ML models will train on the datasets.
3. An AWS Lambda function that processes transactions from the historical datasets and invokes the two Amazon SageMaker endpoints that assign anomaly scores and classification scores to incoming data points.
4. End users (mobile and web clients) invoke Amazon API Gateway REST API for predictions using signed HTTP requests.
5. Amazon Kinesis Data Streams are used to capture real-time event data.
6. AWS Lambda function reads the stream and persists transactional data to a Redis Enterprise Cloud database.
7. AWS Lambda function also leverages Redis Enterprise Cloud as a feature store (No Redis features required for this functionality).
8. AWS Lambda function further persists the prediction results to Redis Enterprise Cloud database. Optionally, the results along with transactional details can also be stored as timeseries data for further data visualizations using Grafana.
9. AWS Lambda function optionally can pass the prediction results via Amazon Kinesis Data Firehose which persists the data to an Amazon S3 bucket so that Amazon QuickSight can consume this data for visualizations and analytics.

Solution Architecture

Redis Enterprise on AWS is a cost-effective, fully managed Database-as-a-Service (DBaaS). This solution takes advantage of the breadth of maturity of the AWS platform services. Additionally, customers can use their financial commitments with AWS towards the purchase of Redis Enterprise Cloud and have one simplified, consolidated billing.

Redis Enterprise has two functions in this solution above. It's been used as a multi-model primary database where it stores and indexes the JSON documents and an online feature store that is being used by SageMaker. A single Redis cluster can serve both these functions. It doesn't have to be separate clusters. AWS Lambda functions consume the messages from Kinesis, apply the rules and sink them to the Redis databases.

Redis Enterprise can also persist data directly into an S3 bucket via periodic backups. In a ML pipeline, this data is the input to train and improve the models running on SageMaker.

The Solution's Primary 5 Components

1. Redis Enterprise as a primary multi-model DB

Redis Enterprise is a powerful and scalable in-memory database that supports advanced data structures. With the addition of its features, Redis Enterprise is a great fit for real-time fraud detection applications.

Let's revisit the rules above.

By using Probabilistic/Cuckoo Filters, we can efficiently implement the blacklisting of the IP addresses.

With Redis commands like GEOSEARCH, GEORADIUS and GEOPOS, it is very convenient to leverage latitude and longitude data.

Redis can store and index JSON objects, via its JSON and Search and Query features.. This, along with the other Redis data types, enables the app to implement the user and purchase profiling logic. Redis also supports time-series data.

This is important if there is a real-time dashboard requirement. Redis Enterprise makes scaling, HA and failover an automated process. This is a huge gain over OSS Redis. What's more, Active-Active and Auto Tiering are Enterprise only features.

Active-Active makes Redis a geo-distributed database and Auto Tiering offers significant cost savings for large databases over OSS Redis.

2. Redis Enterprise as an Online Feature Store

At inference time, the ML engine needs to quickly read the latest feature values from the online feature store and use them at the inference time. Based on the real-time features, the ML model will score the transaction.

Because of its ultra-low latency, Redis Enterprise is a great fit for online and real-time access to the feature data as part of an online feature store solution/implementation.

The Solution's Primary 5 Components

3.
AWS Lambda

With AWS Lambda, you can write a python function that consumes data from Kinesis and sinks back JSON and Time Series data into Redis simultaneously. AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. You can trigger Lambda from over 200 AWS services and software as a service (SaaS) applications, and only pay for what you use.

4.
Amazon Kinesis

In the solution above, we may have parallel consumers, one to sink data to Redis and one to Amazon Sagemaker. Amazon Kinesis is a fully managed service for real-time processing of streaming data at any scale. It provides a serverless platform that easily collects, processes, and analyzes data in real-time so you can get timely insights and react quickly to new information. With Amazon Kinesis, you can ingest data such as video, audio, application logs, website clickstreams, and IoT telemetry data into databases and object stores. Kinesis can handle any amount of streaming data and process data from hundreds of thousands of sources with low latencies.

5.
Amazon SageMaker

The ML layer of our solution is based on Amazon SageMaker. Amazon SageMaker is a complete machine learning (ML) workflow service for developing, training, and deploying models, lowering the cost of building solutions, and increasing the productivity of data science teams. Amazon SageMaker comes with many predefined algorithms. You can also create your own algorithms by supplying Docker images, a training image to train your model and an inference model to deploy to a REST endpoint.

Summary

Fraud rates are at record highs. **Research** from PwC highlights that there's been $42B in losses caused by fraud in 2020 and this figure continues to riseas the numbers of transactions increase.

Maximizing the user experience is crucial and should never be sacrificed when trying to clamp down on fraud. Companies need real-time data and more accurate detection without increasing the latency and implementation costs.

Redis Enterprise is a real-time, in-memory database, which is closely integrated with AWS services, providing users with a cost effective and low latency solution. Below is a quick overview of Redis Enterprise and how it will help you clamp down on fraud:

- Redis Enterprise as a managed service through the AWS Marketplace will allow you to seamlessly scale up (and down) without any drop in performance.
- Reduces cost with high-speed statistical analysis: Probabilistic filters, time series, and other data structures in Redis Enterprise allows you to efficiently check transactions against known patterns before deciding if extensive forensic analysis is needed.
- At inference time, the ML engine, such as SageMaker, needs to quickly read the latest feature values from the online feature store.
- Because of Redis' and AWS's high throughput and low latency services, the solution supports the strict FinServ performance requirements.

Customer reference

Simility is a PayPal service that combines machine learning and human analysis to provide a cloud based fraud detection service. With Redis Enterprise managing billions of transactions per day, Simility was able to make new application functionality 30% faster as well as improving the overall performance by nearly 90%.

Read the full customer story

simility
A PayPal Service

Next Steps

- Check out the Redis Solution Architects AWS Fraud Detection **Github Repo**
- Start using Redis Enterprise Cloud: **AWS MP listing**
- Learn more about Redis and AWS: **redis.com/aws**