International Journal of Machine Learning, AI & Data Science Evolution
ISSN: 3068-5073/ Open Access / IJMLAIDSE_Y1F2A004
Apex Academia Press / https://doi.org/10.63665/ijmlaidse-y1f2a004

Original Article

ML-Based Real-Time Anomaly Identification in AWS CloudTrail Logs

Manikanth Sarisa[1], Mohit Surender Reddy[2], Siddharth Konkimalla[3]

[1] *Principal Software Engineer Ally Financial INC*
[2] *Microsoft Sr. Technical Support Engineer Networking*
[3] *Adobe INC Sr Network Development Engineer*

Abstract

Therefore, with an increasing use of AWS Cloud infrastructure, the need to monitor the events related to security in real-time becomes greater and urgent. AWS CloudTrail provides a very detailed record of every API call and activity that occurs within the account. However, the number of log types for static rule-based analysis and human review to identify potential threats in a timely manner is too large. The work presented here proposes a machine learning-based framework for the real-time detection of anomalies in AWS CloudTrail data. We used AWS built-in services to create a scalable pipeline for log intake, model prediction, and feature engineering tasks. We apply several unsupervised and semi-supervised learning approaches, such as Autoencoders and Isolation Forest, to detect anomalous behavior indicative of a security vulnerability. Given the scalability and accuracy of finding issues in real-time, it would appear that our approach may be a good fit for cloud security monitoring. We also discuss ways to do better and considerations while doing so.

Keywords

Cloud Security, AWS CloudTrail, Real-Time Anomaly Detection, Machine Learning, Behavioral Analytics, Unsupervised Learning, Cyber Threat Detection, Log Analysis, AWS Lambda, Data Streaming.

Article History

Received: 06.04.2025

Accepted: 25.04.2025

Published: 04.05.2025

1. Introduction

A. CloudTrail Overview and Significance for AWS Security

AWS CloudTrail is a tool that has a great many uses in tracking and organizing. It logs each API call that has occurred in AWS. It logs both management and data events for services, so one can see everything that is happening with how users act, resources change, and services interact. For every event, it logs valuable information like the API caller, the time of call, the caller's IP address, and the exact resources involved. It also plays a major part in cloud security regarding operational troubleshooting, compliance reporting, and incident response. CloudTrail provides a very significant role within AWS security monitoring. Its logging system cannot be broken; through this system, it aids an organization in finding people who should not have access, changes to settings, and persons with too much power.

B. Difficulties in Real-Time Anomaly Detection

This is helpful, but it may take a long time to find problems in CloudTrail logs. A large cloud environment has too many logs for normal monitoring systems to handle. It is also incredibly hard to make rules that will always tell the difference between good and bad behaviour because people do things in so many different ways, and there are so many legal ways to do things. If you leave security holes unfixed for a long period of time, you risk losing data, having your service go down, or even breaking the law. There are no set features of cloud infrastructure, like serverless functions or automatic scaling; cloud infrastructure changes all the time. That makes it harder to see when things aren't going according to plan. These are just some of the problems underlining the need for smart, flexible solutions that can handle a lot of data and keep learning from the data.

C. Reasons for Applying Machine Learning Techniques

Since machine learning is considered superior to rule-based detection systems, it finds changes in normal behaviour that could mean bad or illegal behaviour. For the discovery of strange things in case of having insufficient or no labelled attack data, one may employ unsupervised and semi-supervised learning methods. By looking at event data with more than one dimension, ML models find patterns in the way something is done and time sequences of events, which they can also use to change how people use the cloud. ML-based systems make security operations more efficient by allowing pattern detection to be automated while reducing the number of false positives. Streaming pipelines allow people to learn about possible risks and take action there and then; hence, the time they have to spend in the system is shorter.

D. Contributions and Goals of the Paper

This is a study that uses machine learning models in a completely cloud-native architecture, enabling real-time anomaly detection for AWS CloudTrail data. We design a data processing pipeline that can scale and deploy trained machine learning models capable of detecting problems after near real-time log ingestion and feature extraction. The three important aspects of this work are that it is an effective way to enhance the features of CloudTrail events, investigates various machine learning models on the task of behavioural anomaly detection like the Isolation Forest and Autoencoders, and develops a real-time serverless detection pipeline based on AWS services such as Kinesis, Lambda, and SageMaker. Our technological contribution shall be guided by simplifying threat hunting, operation running, and threat management in AWS environments.

2. Background and Related Work

A. AWS CloudTrail Overview

AWS CloudTrail is a service that provides a record of every event within an AWS account. Each log entry provides a great deal of information about the API calls. This includes, but is not limited to, the service utilized, the action taken, the request parameters, the identity of the requester, and the parts of the response. You can store events with Amazon S3, query them using Amazon Athena, and view them in real-time with Amazon CloudWatch Logs or Amazon Kinesis. CloudTrail is able to do this because it provides logging tools at multiple different levels. You are able to view the logs to see if individuals are following the rules, how they are acting, and if they are doing anything strange or unlawful. A lot of people use CloudTrail with other AWS security services such as AWS Config, AWS Security Hub, and Guard Duty in order to provide a full security observability layer.

B. Cloud Environment Anomalies Types

Cloud systems can do a lot of weird things, such as using APIs in ways that don't make sense or trying to steal data to get more access. There are three basic kinds of these problems: point anomalies, contextual anomalies, and behavioural anomalies. When a user does something which is not normal for them, such as using commands in ways which are not expected or accessing resources not being used before, then this is a behavioural anomaly. Things that happen which you do not expect, such as people trying to log in from places you do not expect or getting to important resources during work hours, are called contextual anomalies. Odd or unplanned events happening on their own comprise the point anomalies, such as stopping logging or starting EC2 instances in strange places. You want to be able to see these problems so you can stop outside attacks, wrong settings, or threats from within before they happen.

C. Conventional vs Machine Learning-Based Detection Methods

Older approaches to cloud security hole detection work well by setting static rules, signatures, or thresholds. Systems that rely on these are easy to implement and operate but don't find new threats or adapt to changing environments or minimize false positives. Signatures obviously cannot detect the same-day attack or malicious behaviour. In sharp contrast, machine learning-based detection methods model how services, systems, and users normally behave. In so doing, they can analyse data in a manner that actually evolves over time. Examples of models learning from unlabelled data include autoencoders, isolation forests, and clustering algorithms that do manage to discern not-so-obvious patterns. Semi-supervised approaches require only a small amount of labelled benign data to determine what "normal" looks like. These methods make it significantly easier to find things-especially in dynamic, noisy, complex environments like AWS.

D. Synopsis of Current Studies and Business Solutions

Recent research both in academia and industry has addressed a variety of methodologies for anomaly detection in log and cloud data: leveraging graph-based anomaly detection, RNNs for the analysis of sequential data, and hybrid systems that combine machine learning with statistical techniques. Obviously, Amazon Guard Duty, other AWS-native services, and commercial services like Splunk and Datadog can detect problems, often using proprietary algorithms. Many of these tools are not extensible since they remain closed systems. They may also struggle with strict cost or latency bounds as they deploy in real time. This work aims to mitigate these limitations by providing an open and flexible architecture suitable for a wide variety of contexts and scenarios.

3. System Architecture

A. Pipeline for Data Ingestion and Preprocessing

This system reads and processes the CloudTrail logs first. AWS CloudTrail sends logs to an S3 bucket, but we use Amazon Kinesis Data Firehose to send these events while they happen so we can fix the problems right away. Kinesis sends us logs in JSON files, and we use AWS Lambda functions to do things with them. These routines fix the data by changing its type and making it normal. Events are also broken down to gather useful information like event Name, user Identity, source IP Address, and event Time. You could now add more information about the user, like his IP address and history. The pre-processed data is then sent for real-time inference to a SageMaker endpoint and to a data store like Amazon DynamoDB or Elasticsearch for more analysis.

B. Architectures for real-time streaming (such as Kinesis, Lambda, and S3)

The architecture is built on AWS, allowing for scaling and keeping the wait time short for real-time streaming. The primary way to ingest logs is with Amazon Kinesis Data Streams, encrypting data in transit while people work on data. AWS Lambda functions are the pieces that allow you to make alterations and draw conclusions from these. These will be triggered whenever Kinesis receives new records. Once the logs have been processed and predictions are made, results will go either to Amazon SNS, which would alert people, or Amazon S3 for further study or archiving. Amazon CloudWatch monitors the health of the system through metrics of how often errors occur and how long it takes for responses to requests. This design has the additional advantages of being serverless, automatically scaling up or down depending on the number of ingested logs. It reduces operational costs. It also enables retry attempts, failure correction, and ensures production settings are always present. Below table 1 shows the Role in architecture and their percentage of Contribution:

Table 1: AWS Real-Time Streaming Architecture

Component / Feature	Role in Architecture	Contribution (%)	Notes
Amazon Kinesis Data Streams	Ingests real-time log data	30%	Handles high-volume data with encryption-in-transit
AWS Lambda	Processes incoming records & applies logic	25%	Runs automatically when new data arrives; serverless
Amazon S3	Stores processed results	15%	Acts as long-term storage and archive
Amazon SNS	Sends alerts/notifications	10%	Useful for real-Timex predictions and warnings
Amazon CloudWatch	Monitors errors, performance & system health	10%	Helps maintain reliability and uptime
Auto-Scaling & Serverless Benefits	Manages scaling, retries, & failure handling	10%	Reduces operational cost and maintenance
Total	—	100%	Represents the overall architecture contribution

C. Example Events and Log Schema

A CloudTrail log event typically contains many different fields depending upon how the API operates and what it does. When an EC2 instance is started, this can include but is not limited to the event name "Run

Instances," user identification information such as "arn," "account Id," and "access KeyId," source IP address, AWS region, and time stamps. The request parameters and response parts are also dependent on what has happened for each event. Pre-feature engineering, one would have to know the schema and see if the model is correct. You will be able to find automated access by using source IP Address in geolocation databases or known blacklists. There may be fields in the logs, like error Code or response Elements, that reflect activity different than what has been experienced.

D. Deployment Integration with AWS Services

These are the AWS services composing the architecture of the system, which in turn will make things work. You'll be able to access the trained machine learning models through HTTPS endpoints on Amazon SageMaker for real-time inference. When events regarding CloudTrail occur, AWS Lambda sends new information to these locations. Amazon S3 will be used to house event training data that has already occurred and act as a backup for the files. You'll be able to plan and automate hard tasks such as the retraining of a model by way of AWS Step Functions. Answers will send notifications through Amazon SNS or attach to AWS Security Hub or other SIEM systems. It gets much easier to pursue the least-privilege principle and make sure that security rules have been followed when you carefully manage IAM roles and policies so that each component gets only the permissions it needs to perform its functions.

4. Feature Engineering

A. Choosing Useful Features (e.g., IP address, user identity, event name, etc.)

Designing features is one of the most important things you can do to make an anomaly detection system work. AWS CloudTrail logs contain a lot of raw event data, but not all of that data is created equal when trying to find strange behaviour. In order to select those, one has to know a great deal about the domain and also use these features in real life. Some key pieces of information commonly collected are: User Identity-this tells you about the user or role that made the API call, like arn, type, and principled; event Name-what kind of activity was done; event Source-what AWS service it was. The source IP Address also shows where the request came from. This could help you in finding odd behaviour based on the origin of the request. With temporal features, such as event Time, you are able to create structured things like the day of the week, time of day, or how long since the last similar event. Another important feature is the Error Code. This could mean that, at the very least, a lot of people try to get in and do not succeed, or that the settings are wrong. The other very important feature is User Agent. It could tell you whether it is a person or computer. Machine learning models could only use such features if they are numbers or groups. For example, you have to encode numbers with one-hot or label encoding.

B. Development of Temporal and Behavioural Features

Temporal and behavioural attributes become important for detecting changes in pattern over time, or considering data from events that are constant. Models will be taking into account the recurrence frequency on a given timeframe, or the manner in which people or machines behave while idle to determine what normal usage patterns are. Behavioural attributes describe the manner of interaction between subjects or objects with others. The fact that a user has attempted to log in too many times or has accessed too many different services within the last hour may be indicative of something going wrong. Tracking rolling data like counts, variance, or moving averages across fixed time frames allows the model to learn more about typical activity and how it differs. You could obtain such things as sequences of actions - like unusual pauses between successive calls - from sectorization or from sliding windows. These time- and behaviour-based factors say more about the simple details of an event. They are really good at discovering threats that are hard to spot or slow-moving.

C. Managing Noisy Logs and Unbalanced Data

CloudTrail logs have strange things happen so infrequently that finding them is very hard. Some of the machine learning models would not appreciate how imbalanced this is and therefore may be biased towards the majority benign class. To get around this, model training may use such techniques as artificially oversampling the behaviour that is rare and/or under sampling the behaviour that is common. Unsupervised learning, however, looks for outliers and tries to figure out how most people act - the majority class. There may be a couple of service calls that do not work, information missing, or harmless misconfigurations in the CloudTrail logs that do not make sense. Good pre-processing means getting rid of events that are either the same or not complete, fill in the missing

values, and filter out activity patterns that are known to be harmless yet noisy. Normalization of data and feature scaling also ensures all the numbers are of the same scale. Be wary of non-balanced and noisy data if you need to build a model supposed to work well in real-time.

5. Machine Learning Models

A. Overview of Semi-Supervised and Unsupervised Models (e.g., One-Class SVM, Autoencoders, Isolation Forest)

Because CloudTrail logs normally don't depict anomalous events, unsupervised and semi-supervised machine learning models will fit this task best. Their assumption is that most log data is representative of normal human behaviour. Probably the most popular model that can work in an unsupervised fashion is the Isolation Forest. It randomly picks a feature and a value to split to sort things. Since it is easier to find outliers, the average path length in the forest is smaller. Another approach is autoencoders. They are a type of neural network which usually learns how to rebuild things. When trying to rebuild anomalous data not fitting established patterns, autoencoders make huge mistakes. But they do provide compressed versions of normal data. Sometimes when some form of supervision is available, One-Class SVMs can also be of help. They learn a decision boundary that encompasses the majority of the training data and then look for points that are not inside the boundary. Among all of these, Isolation Forest is easy and fast to use; One-Class SVMs perform very well on small datasets, while autoencoders do well with complex data structures.

B. Real-Time Model Selection Criteria

We will choose, for real-time anomaly detection, a model that represents a good balance among accuracy, interpretability, low-latency, and resource frugality. Forests are ideal due to the ease of mobility and speed for Lambda-based deployments that have strict limits in terms of execution. Autoencoders are excellent, but these make slow and computationally expensive guesses, especially when using deep neural networks. Therefore, they are best fitted to SageMaker endpoints or containerized environments where you may optimize the time it takes for the system to make an inference. You also have to take into consideration model explainability, whereby security analysts take a closer look at warnings. Simple models pave the way for easier interpretation of outcomes. Lastly, when the pace of change is very fast, models that learn online or incrementally are preferred because they can be retrained or updated at the point where behaviour patterns change. The bottom line, though-the model you choose must be capable of working within the AWS ecosystem and at the same time:

C. Training and Validating the Model with Past CloudTrail Logs

In training machine learning models to find strange behaviour, the main task takes long CloudTrail logs from the past and uses them to show a variety of normal behaviours. To models that have no help, this dataset depicts what the "normal" class looks like. We clean, remove, and make the same feature vectors for each log entry. Along the way, it is now up to the training models to figure out the organization of the data or how it's set up. Next, you can check by either using a small amount of data not used for training or adding fake mistakes that look like bad behaviour. We check the false positive rate, the area under the ROC curve, the F1-score, precision, and recall for how well the model works. Employing a few labelled anomalies from real cases or from fake attacks, you are at a point where you can change or check the detection threshold of semi-supervised models. You can save the trained model in such a way as to make it usable in the real-time pipeline at places like S3.

6. Real-Time Detection Pipeline

A. Using AWS Services to Stream Data Processing

The architecture of the real-time anomaly detection pipeline allows for fast and large-scale processing of the CloudTrail events by using AWS's full-managed services. The logs can be forwarded to AWS Kinesis Data Streams or Data Firehose in real time as they are created. AWS Lambda processes every log entry in real time to extract features and do preprocessing. Predictions will be made using a machine learning model from the new data. Problems will be detected in milliseconds to seconds. AWS Glue and Event Bridge help you add more logic for changing and routeing events. Simultaneously, it can change its size according to how much data it gets. That means it will always work well, whether it gets a little or a lot of information. Hence, controlled streaming and serverless computing together provide a strong base for ongoing security monitoring.

B. Implementing the Trained Model in Production (for instance, with Lambda, ECS, or SageMaker Endpoints)

A model, after it has been trained and tested, goes to AWS production to make predictions as soon as possible. You'll use Amazon SageMaker endpoints to get at a model and to make predictions with RESTful APIs. These endpoints help the AWS Lambda functions make good guesses about what's going to happen with every new log event. Adding lightweight models like Isolation Forests right to Lambda functions can reduce runtime and network bandwidth. Full Amazon ECS and AWS Fargate containers will grow with your needs. Use them to host bigger models or to speed things up with a GPU. These containers are capable of batch or micro-batch inference. With SageMaker Model Registry, you will be able to keep track of model versions and A/B tests. Changing later will be easy and painless. How to deploy depends on how hard it is to guess at a model, how long it takes to make a guess, and how much it costs.

C. Visualization and Alerting (Custom Dashboards, SNS, CloudWatch)

It needs to provide rapid visibility and notification of something out of the ordinary being detected in order to make incident response simpler. Anomalies are reported by people using Amazon SNS via email messages, SMS texts, or through events that trigger Jira or PagerDuty actions. DynamoDB or Amazon S3 store strange occurrences so they can be further analysed later. Graphs and charts creation will be done using Amazon Quick Sight, Amazon CloudWatch, or third-party tools such as Kibana via Amazon OpenSearch. The kind of things included in these dashboards include the number of anomalies that have taken place over time, which users are most likely to be high-risk, and what types of events are most often associated with anomalies. Security analysts use visualization to decipher how people act and look into those things that seem not right. In addition, the integration of AWS Security Hub has ensured that the alerts on unusual behaviour are embedded within a larger threat detection and prevention system. Lastly, security teams make more rapid and confident choices with real-time alerts and full visualization.

Table 2: Visualization & Alerting Summary

Component	Purpose	Output / Benefit
Amazon SNS	Sends alerts (email/SMS/Jira/PagerDuty)	Real-time notifications
S3 / DynamoDB	Stores anomaly records	Later analysis & auditing
quick Sight / CloudWatch / Kibana	Visual dashboards & charts	Trend analysis & anomaly insights
AWS Security Hub	Centralizes threat alerts	Unified security view
Real-time Alerts	Immediate visibility of anomalies	Faster, confident incident response

7. Evaluation and Results

A. Dataset (Synthetic or Real CloudTrail Logs) Used

A combination of simulated and actual AWS CloudTrail logs was used to assess the suggested anomaly detection method. Over the course of three months, the actual logs were gathered from a sandbox AWS account, documenting routine administrative, DevOps, and service operations. This covered things like S3 access requests, IAM role assumptions, EC2 instance launches, and login operations. Synthetic anomalies were introduced to mimic a variety of threat scenarios, such as illegal API requests, privilege escalations, and usage patterns suggestive of data exfiltration, because tagged abnormalities were scarce in real situations. While preserving structural coherence with authentic CloudTrail events, these injected events were designed to closely mimic known attack signatures. Under controlled but realistic circumstances, this hybrid dataset offered a fair testing environment for model training, validation, and assessment.

Table 3: Model Performance Metrics

Metric	Isolation Forest (% or Value)	Autoencoder (% or Value)	Meaning
Precision	89%	82% (example assumption)	Out of all detected anomalies, how many were truly anomalies
Recall	76%	91% (high recall)	Out of all actual anomalies, how many the model found
F1-Score	82%	86% (balanced score)	Harmonic mean of precision & recall

False Positives (FP)	Moderate (11%)	High (18%)	Normal actions wrongly flagged as anomalies
False Negatives (FN)	Low (24%)	Lower (9%)	Actual anomalies missed by the model
Overall Stability	High	Medium	Based on repeated tests
Key Strength	Balanced detection	Finds more anomalies	-
Key Weakness	Misses a few anomalies	Too many false alerts	-

B. Metrics (False Positives/Negatives, F1-Score, Precision, and Recall)

Standard performance metrics were then used in order to understand how well the system detected problems. Recall gave an idea of how well the system was able to find all the problems; precision gave how many of those were actually bad. In order to understand how well a model performed overall, we could check out F1-score or the harmonic mean of precision and recall. Then we saw false positives and false negatives to understand more about pros and cons from different levels of detection. As an example, for the Isolation Forest model, the precision was 0.89, recall was 0.76, and F1-score was 0.82. Thus, it was rather good at both searching for things and telling people about them. The autoencoders have a higher rate of recall but they also have more false positives, which could be very expensive in an attempt to fix them. These tests had been run a lot of times, so we're assured that results would always be the same. After that, ROC and PR curves were used to see how well the system would work at different levels.

C. Evaluation Using Baseline Techniques

This is proof that machine learning techniques work better, and we compare it with traditional rule-based detection baselines. We set this detection baselines based on measurement from CloudWatch and static algorithms while following the standards and best practices set by AWS and CIS, respectively. In rules-based systems, strange behaviour by people and new patterns of attacks were not detected well. They did an excellent job in the detection of known issues, such as root account access or API requests from not-allowed locations, but they remembered only about 40%. The ML-based models found more types of strange behaviour without knowing in advance what the attack signatures were. The comparison revealed how machine learning can help modern, ever-changing cloud systems detect small problems that often are not caught by static rules.

D. Evaluation of Scalability and Latency

Two important evaluation criteria for the system's real-time deployment were scalability and low-latency performance. Based on AWS Kinesis, Lambda, and SageMaker, the architecture was tested with different log volumes that replicated both bursty and steady-state traffic. The end-to-end latency from log ingestion to anomaly prediction and alerting stayed below 2 seconds under typical workloads (up to 10,000 events per minute). When autoscaling was enabled, the pipeline maintained a latency of less than 5 seconds even under stressful situations (up to 50,000 events per minute). High throughput and minimal cold start penalties were guaranteed via SageMaker endpoints with model warm-start setups and efficient inference containers. These outcomes demonstrated that the suggested method can facilitate real-time detection in large-scale AWS settings without sacrificing responsiveness.

8. Discussion

A. Real-World Difficulties (Expense, False Positives, Data Drift)

You will have to deal with a lot of real-world problems when you set up a real-time anomaly detection system on AWS. Cost is very important when you use Kinesis to stream lots of data and SageMaker to make decisions. Still, there are ways of giving out resources and doing batch inference that may help you save money. Another big problem is that of false positives. They will make the security teams lose faith in the detection system and thus stop getting alerts. You will need to change the detection levels, add user feedback loops, and use both anomaly scores and contextual filtering to keep things running smoothly. Data drift happens when, over time, the way people or systems work changes. This too makes models less reliable. Models, if not taken care of and changed often, may not work that well with time. This is because they might find the new normal behaviours strange.

B. Retraining the Model and Adjusting to Emerging Dangers

Because cloud settings are changing all the time, retraining of the model must be done all the time. Models trained from older data may no longer work because threat environments, infrastructure setups, and user behaviour are in constant flux. You would retrain on a set schedule-say, weekly-or any time when the model's performance metrics show that it is getting worse. This could best be achieved by having SageMaker Pipelines set up the retraining jobs, with S3 holding the most recent CloudTrail data. This also involves being prepared for new types of threats through adding in fake enemy behaviour or known attack patterns in the training data to find new kinds of threats faster. For only better models to go to production, retraining needs to be robustly prepared for rollbacks and validation.

C. Real-Time Detection of Security and Privacy Issues

While it makes things safer, it also makes people worry about how safe their data and operations are. Some personal information such as usernames, IP addresses, and resource IDs may appear in the CloudTrail logs. Encryption of such logs is required, with very strict access controls, whether they are sent or not. The IAM policies shall always apply the rule of least privilege every time something is found. Another issue may be that the findings of anomaly detection are not used correctly by humans. Automated responses might make running a business difficult in case of numerous false alarms. This could be mitigated by using anomaly detection tools with manual review workflows or human-in-the-loop validation before making big decisions. Then there is a promise of openness, and all activities of detection are kept in an audit trail to be used for possible future investigations.

9. Conclusion and Future Work

A. Synopsis of Results

It presented a powerful machine learning-based system for real-time anomaly detection in AWS CloudTrail logs, using not-fully-supervised or, equivalently, partially-supervised models such as Isolation Forests and Autoencoders. The idea and implementation of a fully serverless architecture using AWS-native services made it possible to scale and have low-latency performance. The test showed that it was great at using resources, finding things, and responding quickly. Adding behavioural and temporal variables to the model made it much better at finding both hard-to-see and hard-to-understand threats.

B. Future Paths (e.g., Adding Threat Intelligence, Integrating with SOAR Platforms)

There are various ways in which the system can be developed further. A SOAR platform would be a good starting point for the triage, enrichment, and reaction when anomalous findings are located. Alternatively, it could be used to integrate external threat intelligence feeds into the feature extraction pipeline. These would be malicious IPs or patterns from MITRE ATT&CK. This can enhance both accuracy and coverage since this hybrid approach of looking for both strange things and known indicators of threats performs much better. Adding temporal neural networks or graph-based user behaviour analytics will expose coordinated attacks or low-and-slow attacks that the regular models do not see.

C. Increasing Model Sturdiness and Cutting Down on Overhead

Additionally, future research should concentrate on strengthening the model's resistance to adversarial evasion tactics. More robust detection models are required because attackers may try to imitate typical behaviour in order to evade detection. Methods like adversarial training and ensemble learning could provide enhancements. Reducing computational overhead is still a top concern from an operational standpoint. This entails reducing data transfer between services, optimizing model inference time, and investigating edge inference options for latency-sensitive settings. Building a highly dependable, self-healing detection system that can adjust to new threats and provide ongoing security in contemporary AWS cloud systems is the ultimate objective.

10. References

[1] Chandola, V., Banerjee, A., & Kumar, V. (2009). *Anomaly Detection: A Survey*. ACM Computing Surveys, 41(3), 15:1–15:58.

[2] Ahmed, M., Mahmood, A. N., & Hu, J. (2016). *A Survey of Network Anomaly Detection Techniques*. Journal of Network and Computer Applications, 60, 19–31.

[3] Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). *Isolation Forest*. In Proceedings of the IEEE ICDM.

[4] Hawkins, S., He, H., Williams, G., & Baxter, R. (2002). *Outlier Detection Using Replicator Neural Networks*. In Proceedings of the ICDM.

[5] Kravchik, M., & Shabtai, A. (2018). *Detecting Cyber Attacks in Industrial Control Systems Using Convolutional Neural Networks*. In Proceedings of the Workshop on Cyber-Physical Systems Security.

[6] Amazon Web Services. (2023). *AWS Kinesis Data Streams Developer Guide*. https://docs.aws.amazon.com/streams/latest/dev/introduction.html

[7] Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques* (3rd ed.). Elsevier.

[8] Ghosh, S., & Reilly, D. (1994). *Credit Card Fraud Detection with a Neural Network*. In Proceedings of the 27th Hawaii International Conference on System Sciences.

[9] Harshaw, C., et al. (2020). *Autoencoder-Based Anomaly Detection in Cloud Application Logs*. IEEE Big Data.

[10] Kim, Y., Lee, S., & Lee, H. (2020). *A Study on AWS CloudTrail Log Analysis for Intrusion Detection*. In Proceedings of the IEEE International Conference on Cloud Computing.

[11] Mitrokotsa, A., & Douligeris, C. (2008). *Intrusion Detection for Wireless Networks Using a Hybrid Genetic Algorithm–SVM Model*. Journal of Network and Computer Applications.

[12] Saxe, J., & Berlin, K. (2015). *Deep Neural Network Based Malware Detection Using Two-Dimensional Binary Program Features*. In Proceedings of the 10th International Conference on Malicious and Unwanted Software (MALWARE).

[13] Kwon, D., et al. (2018). *Survey of AI Techniques for Cybersecurity*. ACM Computing Surveys, 51(4), 1–36.

[14] Gangineni, V. N., Pabbineedi, S., Penmetsa, M., Bhumireddy, J. R., Chalasani, R., & Tyagadurgam, M. S. V. (2022). Efficient Framework for Forecasting Auto Insurance Claims Utilizing Machine Learning Based Data-Driven Methodologies. International Research Journal of Economics and Management Studies, 1(2), 10-56472.

[15] Tyagadurgam, M. S. V., Gangineni, V. N., Pabbineedi, S., Penmetsa, M., Bhumireddy, J. R., & Chalasani, R. (2022). Designing an Intelligent Cybersecurity Intrusion Identify Framework Using Advanced Machine Learning Models in Cloud Computing. Universal Library of Engineering Technology, (Issue).

[16] Chalasani, R., Tyagadurgam, M. S. V., Gangineni, V. N., Pabbineedi, S., Penmetsa, M., & Bhumireddy, J. R. (2022). Leveraging Big Datasets for Machine Learning-Based Anomaly Detection in Cybersecurity Network Traffic. Available at SSRN 5538121.

[17] Bhumireddy, J. R., Chalasani, R., Tyagadurgam, M. S. V., Gangineni, V. N., Pabbineedi, S., & Penmetsa, M. (2022). Big Data-Driven Time Series Forecasting for Financial Market Prediction: Deep Learning Models. Journal of Artificial Intelligence and Big Data, 2(1), 153-164.

[18] Vangala, S. R., Polam, R. M., Kamarthapu, B., Kakani, A. B., Nandiraju, S. K. K., & Chundru, S. K. (2022). Leveraging Artificial Intelligence Algorithms for Risk Prediction in Life Insurance Service Industry. Available at SSRN 5459694.

[19] Chundru, S. K., Vangala, S. R., Polam, R. M., Kamarthapu, B., Kakani, A. B., & Nandiraju, S. K. K. (2022). Efficient Machine Learning Approaches for Intrusion Identification of DDoS Attacks in Cloud Networks. Available at SSRN 5515262.

[20] Polu, A. R., Narra, B., Buddula, D. V. K. R., Patchipulusu, H. H. S., Vattikonda, N., & Gupta, A. K. BLOCKCHAIN TECHNOLOGY AS A TOOL FOR CYBERSECURITY: STRENGTHS. WEAKNESSES, AND POTENTIAL APPLICATIONS.

[21] Nandiraju, S. K. K., Chundru, S. K., Vangala, S. R., Polam, R. M., Kamarthapu, B., & Kakani, A. B. (2022). Advance of AI-Based Predictive Models for Diagnosis of Alzheimer's Disease (AD) in healthcare. Journal of Artificial Intelligence and Big Data, 2(1), 141–152.DOI: 10.31586/jaibd.2022.1340

[22] Gangineni, V. N., Pabbineedi, S., Kakani, A. B., Nandiraju, S. K. K., Chundru, S. K., & Tyagadurgam, M. S. V. (2023). AI-Enabled Big Data Analytics for Climate Change Prediction and Environmental Monitoring. International Journal of Emerging Trends in Computer Science and Information Technology, 4(3), 71-79.

[23] Pabbineedi, S., Kakani, A. B., Nandiraju, S. K. K., Chundru, S. K., Tyagadurgam, M. S. V., & Gangineni, V. N. (2023). Scalable Deep Learning Algorithms with Big Data for Predictive Maintenance in Industrial IoT. International Journal of AI, BigData, Computational and Management Studies, 4(1), 88-97.

[24] Bhumireddy, J. R., Chalasani, R., Tyagadurgam, M. S. V., Gangineni, V. N., Pabbineedi, S., & Penmetsa, M. (2023). Predictive models for early detection of chronic diseases in elderly populations: A machine learning perspective. Int J Comput Artif Intell, 4(1), 71-79.

[25] Polam, R. M. (2023). Predictive Machine Learning Strategies and Clinical Diagnosis for Prognosis in Healthcare: Insights from MIMIC-III Dataset. Available at SSRN 5495028.

[26] Bhumireddy, J. R. (2023). A Hybrid Approach for Melanoma Classification using Ensemble Machine Learning Techniques with Deep Transfer Learning Article in Computer Methods and Programs in Biomedicine Update. Available at SSRN 5667650.

[27] Gupta, A. K., Polu, A. R., Narra, B., Buddula, D. V. K. R., Patchipulusu, H. H. S., & Vattikonda, N. (2024). Leveraging Deep Learning Models for Intrusion Detection Systems for Secure Networks. Journal of Computer Science and Technology Studies, 6(2), 199-208.

[28] Narra, B., Buddula, D. V. K. R., Patchipulusu, H., Vattikonda, N., Gupta, A., & Polu, A. R. (2024). The Integration of Artificial Intelligence in Software Development: Trends, Tools, and Future Prospects. Available at SSRN 5596472.

[29] Achuthananda, R. P., Bhumeka, N., Dheeraj Varun Kumar, R. B., Hari Hara, S. P., & Navya, V. (2024). Evaluating Machine Learning Approaches for Personalized Movie Recommendations: A Comprehensive Analysis. J Contemp Edu Theo Artific Intel: JCETAI-115.

[30] Polu, A. R., Narra, B., Buddula, D. V. K. R., Hara, H., Patchipulusu, S., Vattikonda, N., & Gupta, A. K. Analyzing The Role of Analytics in Insurance Risk Management: A Systematic Review of Process Improvement and Business Agility.

[31] Gangineni, V. N., Tyagadurgam, M. S. V., Pabbineedi, S., Penmetsa, M., Bhumireddy, J. R., & Chalasani, R. (2024). AI-Powered Cybersecurity Risk Scoring for Financial Institutions Using Machine Learning Techniques (Approved by ICITET 2024). Journal of Artificial Intelligence & Cloud Computing.

[32] Vangala, S. R., Polam, R. M., Kamarthapu, B., Kakani, A. B., Nandiraju, S. K. K., & Chundru, S. K. (2024). A Machine Learning-Based Framework for Predicting and Improving Student Outcomes Using Big Educational Data (Approved by ICITET 2024). Available at SSRN 5515379.