National College of Ireland

AI-Driven Self-Learning Intrusion Detection & Response System for Cloud Security

MSc Research Project
MSc in Cybersecurity

Rohan Vivek Lahane
Student ID: x23251506

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Rohan Vivek Lahane
Student ID:	X23251506
Programme:	MSc in Cybersecurity **Year:** 2024-25
Module:	Practicum
Supervisor:	Michael Pantridge
Submission Due Date:	11-08-2025
Project Title:	AI-Driven Self-Learning Intrusion Detection & Response System for Cloud Security
Word Count:	6322 **Page Count:** 20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Rohan Vivek Lahane

Date: 11-08-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

AI-Driven Self-Learning Intrusion Detection & Response System for Cloud Security

Rohan Vivek Lahane
X23251506

Abstract

Sophisticated cyber threats in cloud environments, especially those that are hosted on platforms such as Amazon Web Services (AWS) are shedding more and more since they are complex, and scalable in nature as well as their dynamic user access model. The current types of attacks on clouds that exploit identity and access management misconfigurations are quite difficult to be tracked with traditional intrusion detection systems (IDS). The present project suggests an AI-based, self-educating intrusion Detection and Response System (IDRS) that uses both supervised and unsupervised machine learning (XGBoost, Isolation Forest) in detecting abnormalities in the behaviour of cloud traffic.

Locally, the system is prototyped with cleaned KDD datasets and simulated AWS-like CloudTrail logs to mimic the real-world clouds, and their activities with unauthorized access, privileged escalation, and reconnaissance attempts. The solution has proven the power of using both AI models and simulated identity-driven detection of threats to secure AWS-style environment in the case that there is no direct cloud connection. The results of the model are improved through the measurement of the performance of the model in terms of accuracy, precision, recall, and confusion matrices.

This paper establishes a fault-tolerant future deployment of AWS by suggesting how it is possible to translate this system into consuming genuine CloudTrail logs, generating automatic response, and interfacing with SOAR solutions. Although locally developed, the architecture is entirely scalable to a cloud-native environment, and opens the door to automating response and continuous learning in contemporary cyber defence.

1. Introduction

The high rate of migrating businesses and important infrastructure to the cloud including Amazon Web Services (AWS) has made the threat landscape huge. With the popularization of cloud computing in which organizations store their sensitive data, manage identities of their users, and help them dynamically scale applications, hackers are retooling their tactics to take advantage of cloud peculiarities. Conventional intrusion detection systems (IDS) developed to support a more static and on-premise environment have proven to be insufficient at providing complexity and scale to cloud-native environments. As this widening range suggests, there should be greater emphasis on using intelligent, dynamic and responsive security measures to identify security threats as well as combat them in real time in cloud environments.

Among the major issues of cloud security are improper exploitation of identity and access management (IAM) controls, including security-bypassing roles, credential misuse, or privilege escalation through poorly designed control policies. Under the MITRE ATTACK, the attacks implemented by many cloud services currently focus on the authentication and authorization mechanisms and not only the network endpoints. The current suppliers of cloud services provide security tooling; examples of which are; AWS Guard Duty and CloudTrail; these tools can be limited to a set of rules/Configurations or contain static thresholds, etc. such that they are ineffective in detecting new or obfuscated attack patterns.

This study delivers an AI-generated self-learning Intrusion Detection and Response System (IDRS) which uses both supervised and unsupervised (XGBoost, Isolation Forest) machine learning models to identify anomalies in cloud style activity. The system under development and testing is designed to be run in reality AWS environments in implementation. The system has publicly available data (e.g., NSL-KDD) and synthetic logs generated internally to represent IAM abuse, unauthorized access, and behaviours that deviate from the norm in cloud services to emulate the behaviour of AWS.

The major research question under which the current project unfolds is the following one:

Is it possible to remotely detect suspicious or unauthorised activity, based on learned model trained on structured and cloud-representative activity data using the AI, and simulate a system that responds to the suspicious and unauthorised activity in real time with the proper infrastructure to use on AWS?

This research aims at achieving the following objectives:

- To build and develop a supervised (XGBoost) and unsupervised (Isolation Forest) AI-based IDS.
- To recreate the type of threats found in a cloud (i.e. IAM role misuse, malicious API invocations) by synthesizing logs that are similar to the AWS CloudTrail logs.
- In order to compare the success of the system on the basis of classification measures (precision, recall, F1-score, confusion matrix).
- To be able to move the system to the cloud at some point in the future, it is necessary to diagram the architecture and components of the system and the corresponding matching AWS service.

The study will fill in the gap in the scientific literature as the research dramatically demonstrates the possibility of simulating and printing dynamic, adaptive solutions to IDS based on AI on cloud environments without the mandatory availability of cloud sources. It demonstrates the way of modeling real-world scenarios using available open-source tools and structured datasets, and provides a plan of further extending the solutions to real-world live cloud systems powered by use of AWS-native services.

2. Related Work

The given section provides an overview of the available academic sources on intrusion detection systems (IDS), focusing specifically on the AI-powered ones, their applicability to a cloud environment, and, more specifically, Amazon Web Services (AWS). Reviewed methods are both supervised and unsupervised, as well as the application of Generative Adversarial Networks (GANs), anomaly identification, and AWS-specific security concerns related to IAM. This will be aimed at critically evaluating the limitations and contributions made by the previous studies, and the rationale of conducting the proposed study.

2.1 The Machine Learning and AI in IDS

Many researchers have worked on how AI and machine learning can be used in intrusion detection. Recently, Jeyanthi and Tiwari (2023) proposed an AI-based IDS cloud environment based on GANs and mentioned higher detection rates due to data augmentation with the synthesis of data. Nevertheless, the implementation did not put emphasis on IAM-vulnerable attacks.

- Li et al. (2022) recently made a proposal on MAD-GAN, which was an idea on multivariate anomaly detection on time-series data. The method did also effective in ICS environments, however, it could not have been directly applied to such environments cloud traffic as AWS CloudTrail. On a comparable note, Ullah and Mahmoud (2020) presented a hybrid IDS called G-IDS that uses GAN and CNN and reports high performance on the NSL-KDD data set, however, without cloud-related adaptation.

- To benchmark the IDS robustness, Lin et al. (2018) came up with IDSGAN-produced adversarial traffic. It was a breakthrough in adversarial learning but it did not consider the mitigation-driven response models.

- The most effective classifier is the one that was assessed in a comparative investigation by Aljawarneh et al. (2018) based on NSL-KDD, which is ensemble learning. Although useful, the data itself is outdated, and cloud-native data (e.g., CIC-IDS2018) are more applicable.

- Yousef et al. (2023) considered available datasets and methods of detection. Their main point was that little labelled cloud-specific intrusion data exist, which justified augmentation through GANs.

2.2 Cloud and AWS IDS

- Sharma and Sharma (2022) suggested a mixture model based on fuzzy clustering and autoencoders to conduct intrusion detection on AWS based on CIC-IDS2018. Although the results were encouraging, they were evaluated using hyperparameters that were manually adjusted, and IAM abuse cases were not investigated.

- By using temporal data, Rahman et al. (2023) paid special attention to anomaly-based detection of intrusion into the cloud environment. Their model was explainable but it

was correct. By the same token, with the LightGBM approaching GAN-generated data, Li and Wang (2022) provided little information about the real-time deployment.

- Arguing that cloud IAM can conceptually be connected with intrusion detection on an identity level, Dehghantanha et al. (2016) introduced the idea of an identity-centric intrusion detection mechanism to the IoT. Nevertheless, they did not work on AWS.

- Likewise, on a more general survey, Osanaiye et al. (2016) discussed identity-based threats in cloud, citing IAM misconfigurations as one of the primary threat vectors. That is in line with our interest in IAM vulnerabilities.

- Almseidin et al. (2017) have implemented the particle swarm optimization in deep belief networks to detect intrusions in clouds and improved both converging accuracies. Nonetheless, there were no metrics of implementation as to scalability.

2.3 AWS Vulnerabilities of Identity and Access Control

The study followed a study that has analyzed the AWS ecosystem in terms of identity. Alharkan and Martin (2020) offered an understanding of IAM abuse scenarios to determine how such abuse unfolds (e.g., excessively permissive roles, and session hijacking) and how these instances are aligned with MITRE ATT&CK methods.

- The research by Liu et al. (2021) focuses on AWS Cognito and S3 ACLs misconfiguration that demonstrates that insecure defaults may result in privilege escalation. Their studies make it evident why automated IAM audit solutions are needed.

- Neves et al. (2022) have pointed out tools such as ScoutSuite, CloudFox, and Prowler as effective to use when performing AWS reconnaissance and identify misconfigurations. They are however rule based and not adaptively intelligent.

- Lastly, Mohamed et al. (2023) suggested threat modeling aimed at IAM in line with MITRE ATT&CK, another one of the cornerstones of our proposed framework.

2.4 Summing-up

The literature shows the evident acceleration of the innovations progress concerning ML- and GAN-based intrusion detection systems, yet currently, there is a lack of it concerning the cloud-native environments, such as AWS. Present IDSs are usually network-centered or erected upon age-old datasets such as NSL-KDD. Few efforts have been made of merging IAM policies and the detection of adversarial behavior in AWS. Thus, an identity-sensitive, GAN-employing IDS model to support AWS has a high potential need with the use of real cloud records like CloudTrail, which is the target of the presented study.

3. Research Methodology

This segment describes the overall approach followed during the process of development and testing of an AI-based Intrusion Detection System (IDS) used to provide an improved level of cloud security, in particular, in AWS environments. The approach is meant to be non-replicable, transparent, and based both on the realistic needs of cybersecurity and well-established scholarly study.

3.1 Overview Research Design

This study uses a design science approach where the researcher will aim to develop a machine learning-based Intrusion Detection System (IDS) specialized to detect threat incursions on cloud computing environments; specifically in AWS. The drive is to fill the knowledge-practice gap concerning theoretical and practical security requirements in cloud computing. The main goal is to automate the intrusion detection in dynamic and scale able cloud environment which simulates data traffic much like a real-world scenario and it measures the performance of the system with controlled conditions. It involved a hybrid strategy where the experimental implementation, the simulation of cloud characteristics, and an abusive analysis of machine learning models have been combined.

3.2 Preprocessing and data collection

The work is based on using the KDDCup 1999 dataset, one of the old datasets related to network intrusion detection. Even though old, this dataset has a vast amount of labeled data that can be used in prototyping machine learning eco-models. KDDTrain_cleaned.csv and KDDTest_cleaned.csv were used to split the dataset in- to a training and a test set with 41 features per entry of traffic information about the network, like protocol type, service, or duration and flag. Label encodings were applied to categorical columns such as protocol_type, service and flag. A standard scaler was then used to normalize the features so that they are distributed uniformly. The labels were converted to be of two classes where normal was assigned to 0 and all categories of attacks were assigned to 1.
Besides the KDD dataset, some simulated AWS CloudTrail records (sample_cloudtrail_logs.json) were generated to have the resemblance to the live AWS usage patterns. The purpose of these logs was to simulate normal and abnormal cloud activities, e.g., IAM role abuse, unauthorized API requests, and reconnaissance activity. The recorded events had important attributes of the CloudTrail such as eventTime, eventName, sourceIPAddress, and identity information of the users which are frequently utilized in intrusion detection on the cloud.

As a method of live-testing, these simulated logs could be run through the same preprocessing pipeline used on the KDD dataset:

- Fields that should be encoded assigning a category to a non-numeric value (e.g. protocol type, event name).
- Scaling features into normalcy of the numeric features.
- Label categorisation: 0 to normal activity and 1 to suspicious/ attack activity.

In near-real time, the trained XGBoost model was passed the processed events and the output prediction was written onto predicted_live_logs.csv that was the input data source to the Live Intrusion Detection Streamlit Dashboard. In this dashboard, it was possible to constantly observe the outcomes of detection, summary of suspicious events as well as a trend analysis over time.
An offline predictionary log was kept as a record of all historical predictions in a separate file in the form of predicted_logs.csv which allowed offline assessment, comparisons of performance and occasional retraining using newly annotated data. These two configurations: real-time streaming to allow operations awareness and off-line storage to provide retrospective analysis ability enabled it to test the system in more of an AWS-like environment but maintain analytical capabilities.

3.3 Development of models

There were two kinds of models that have been put in place on the comparison between the supervised and unsupervised learning in intrusion detection. The supervised learning method used XGBoost classifier because of its speed and performance using the structured data. It has the capacity to work with big datasets and noise and class imbalance. In the case of unsupervised learning, the Isolation Forest was applied in the anomaly detection, which imitated a situation when unusual patterns of attacks appear. The development and training of both models were done in Python along with Libraries like scikit-learn and XGBoost. The possibility of relying on deep learning features, including Generative Adversarial Networks (GANs) at a later version was also provided.

3.4 Implementation Environment and Setup Done with Experiment

The development and all the experiments were done locally via Visual Studio Code within a Windows 11 system of 16 GB RAM with an AMD Ryzen 7735HS chip. A dependency environment was created so as to maintain clean dependencies. Even though the study is AWS cloud security-oriented, it tested its models offline since accessibility to cloud infrastructure has limited access. Nonetheless, the system was formulated in a way that it can expand to take account of AWS-specific combinations of logs like CloudTrail, IAM policy settings and real-time EC2 network activities.

3.5 Methodology of Evaluation

Model evaluation was performed on trained-test split of 70-30 that ensures even distribution of normal and attack classes ratios. Some of the common classification measures were used to quantify the performance; that is, accuracy, precision, recall and F1-score among others. These metrics provided the possibility to compare the extent to which each of the models is able to identify attack patterns and prevent the false positive. Confusion matrices and precision-recall curves were plotted to have a visual representation and perform analysis through Matplotlib and Seaborn libraries. Domain knowledge was also used to filter and analyze false positives e.g. by marking known benign services or IPs that would otherwise cause them to be misclassified. This assisted in the streamlining of a model making decision.

3.6 Statistics and Charting

These were analyzed graphically as bar equations, and confusion matrices to show how many of these are correctly or incorrectly classified. Its results were compared to the results of both supervised and unsupervised model, and it was noticed that even though the supervised model had higher accuracy, the unsupervised model was superior at identifying unknown attacks. This kind of visual feedback proved critical in determining the feasibility of the models and where efforts should be directed towards an improvement. Such visuals played a big role in the justification of the necessity of incorporating the model outputs with AWS services or security dashboards/Enterprise to arrive at kinetic actions.

3.7 Limitations and Assumptions

The current version was made to operate on Amazon Web Services (AWS) but it was done on a dormant AWS environment and not operational in a live production-ready environment but locally on a simulated version. Although suitable to testing proof-of-concepts, the procedure failed to subject the system to the variability and contingency of real-world cloud set ups. The employed dataset is somewhat comprehensive, but does not reflect the properties of contemporary cloud-native traffic and the complexity of advanced persistent threats, and the quality of model performance depends on the quality of the dataset and feature engineering.

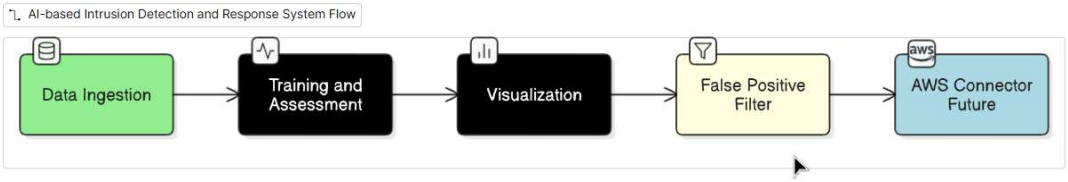
High-quality sources of cloud-native telemetry (like AWS CloudTrail logs, or VPC Flow Logs, or Amazon S3 access patterns) were listed as possible additions to the future iterations to enhance the realism and the detection accuracy. Consequently, the findings are only considered as a conceptual representation, a scalable base that can be tuned up with live cloud, more detailed data and expansive telemetry going forward.

4. Design Specification

The suggested solution would be a modular and AI-based Intrusion Detection System (IDS) to be simulated to and ultimately deployed to a cloud environment, AWS in this case. The task is to identify malicious or abnormal activity in the data of the network traffic by using both learning models supervised and unsupervised types of machine learning. Although the current implementation is offline, and uses cleaned KDDCup99 datasets, the architecture has been implemented to be used with AWS CloudTrail logs, VPC Flow Logs, as well as other telemetry data.

4.1 Architectural Design

System is designed into the following modular structures:



1. **Data Ingestion & Preprocessing Module**

This module will process raw CSV files (which were initially KDDTrain/KDDTest), convert labels to binary classification (in case those fields are categorical), normalize numbers and convert the number to label. It has a role in changing received data into the form suitable to train and infer.

2. **Training/ assessment Element Table**

Two models are trained, and tested:

- Supervised Model: XGBoost classifier, a model that is trained with labelled and offers a high accuracy of attack classification.
- Unsupervised Model: Isolation Forest: detects, without labelling, anomalies, and can be useful with zero-day or novel threats.

3. **Visualization Module**

Visualizes metrics provided by matplotlib and seaborn, e.g. confusion matrices, ROC curves, feature importance rankings.

4. **False Positive Filter**

Rules based filter effectively checking against a whitelist of known harmless sources (e.g. internal IPs, trusted services) of intrusions.

5. **(Future Integration) AWS Connector Layer**

Optimised to connect it to real AWS services with the aim of detecting it and automating its mitigation, such as CloudTrail, CloudWatch, GuardDuty and AWS Lambda.

4.2 Description of Model and Algorithm

- **Supervised Learning Algorithm XGBoost**

This algorithm is chosen because it is robust to tabular data, and its ability to process imbalanced classes: XGBoost (Extreme Gradient Boosting). It constructs a series of decision trees and tries to improve the mistakes made by the previous tree in a gradient descent optimisation procedure. The training procedure aims at trying to reduce a regularized loss function to avoid overshooting. It is compatible with aborted, regularization and parallel computation.

- **Unsupervised Learning Algorithm -IsolationForest**

Isolation Forest algorithm is based on the fact that it is easier to isolate anomalies than normal points. It builds various binary trees with each of the splitting being done on a randomly selected feature and a randomly selected split value. Anomaly scores are determined using the average length of the path of all of the trees to a given point. Locations with low paths are regarded to be outliers.

4.3 Design Requirement

- Scalability: The model must be scalable to real time data sources at AWS cloud.
- Portability: Python code, can be deployed as a local script or a Lambda in AWS.
- Modularity: This is done so that each of the modules (preprocessing, model training, evaluation) can be isolated easily to be tested and improved upon separately.
- Security Integration: Ability to integrate with AWS security services such as IAM, GuardDuty and CloudTrail in order to detect unwarranted access.

5. Implementation

The last execution of the AI-Driven Intrusion Detection System (IDS) was also carried out based on the Python programming language, with the use of the modular design which has already been explained above. The implementation process was aimed at the processing of pre-processed datasets into an organized form to serve as inputs to machine learning models, training the models and quantitative measurement of the precision of learning processes. This section presents the main building blocks that are applied, the tools and libraries that are in use and outputs.

5.1 Development System, Toolset

We executed the project on the Visual Studio Code integrated development environment (IDE), where we had a specially created Python virtual environment (.venv) to maintain the dependencies. The parts were all written using Python 3.9. These key libraries and frameworks were applied:

To handle the data and do the calculations, pandas and NumPy will be used.

- Scikit-learn to perform data preprocessing, train (Isolation Forest) and assess evaluation metrics.
- XGBoost supervised classification with gradient boosting.
- Performance measure visualization and feature analysis by Matplotlib and Seaborn.
- Joblib to keep and reuse models.

Its source was structurally split into modular segments, where separate Python modules were composed of data preprocessing, model training and evaluation, and visualizations. The project lay out was aligned to clean software engineering concepts so that it could be maintained and extended or even restructured into cloud systems like AWS.

5.2 Artifacts and Outputs

In incorporating what would happen, the implementation resulted in some core outputs that were necessary to validate the intrusion detection system. The former was the conversion of raw data to a usable one. Data preprocessing has been applied to the original datasets (KDDTrain_cleaned.csv and KDDTest_cleaned.csv) various transformations, e.g., label encoding to categorical attributes and scaling with StandardScaler the numerical attributes as normalization. This labelled data was further divided into features and labels so that supervised and unsupervised models could equally be trained successfully.

There were two models developed and tested, XGBoost classifier supervised learning algorithm to predict classes of attack depending on the input features and the Isolation Forest unsupervised anomaly detection tool that was employed to identify anomalous network activity behavior. The XGBoost and the XGBRT models were trained and tested and the XGBoost model performed better in terms of precision and recall measure. The models were serialized with the help of joblib in order to be persistent and used in the future.

Also, measures of evaluation and graphical results were produced in favour of analysis and interpretation. These consisted of classification reports, confusion matrices and feature importance plots. In order to garner refinements in detection accuracy, false-positive elimination procedure was introduced using benign IP and prototypes working, which aided in emulating a more reasonable intrusion response mechanism.

5.3 Prediction in Real-time on Simulated CloudTrail Logs

A live prediction workflow was implemented in the system, to fill the gap between the offline model evaluation and an operational setting of the intrusion detection scenario. This functionality utilizes simulated AWS CloudTrail data to allow them to simulate the constant security events that would be received in a live cloud environment. A special data set was generated sample_cloudtrail _logs.json, to resemble the format of genuine AWS CloudTrail logs. These fake logs contain a combination of benign actions, like valid API requests or account logins, with signals of suspicious behaviour akin to privilege escalation, or policy abuse. In doing so, the data is organized to present an accurate and difficult to detect scenario within the live testing environment whilst not reliant on direct connectivity with AWS.

The simulated events are conducted through the same preprocessing and feature engineering procedures that are applied to conduct the training of models. This guarantees the input to the XGBoost (XGBoost trained) classifier to be consistent with regards to format/scale, which is essential in ensuring reliability once prediction is conducted. After processing each incoming record falls into one of the categories; normal or an attack and the prediction alongside the confidence score is added to a file called predicted_live_logs.csv. The file indicates the active level of the threat evaluation of the system at any particular time. Predictions are stored in predicted_logs.csv that add results of multiple separate sessions to compute historic tracking and trends.

To allow checking the real-time prediction process and monitor the accuracy, a Streamlit dashboard (view_live_predictions.py) was created. It reads in the most recent predictions file and outputs the results in an easy to use, interactive dashboard. It contains the visual statistics of the counts of predictions, the time-based trends of merely found anomalies, as well as specific tables that depict the most recent events and the resultant predictions. The user has the option of only filtering certain types of attacks by clicking on a drop-down option in order to analyse those attacks. Despite the fact that it is driven by simulated data this live dashboard displays how the IDS might be implemented in the cloud-based intrusion detection system, which guarantees constant situational awareness and enables the proactive incident response.

6 Evaluation

6.1 Introduction

This section gives an in-depth analysis of this intrusion detection model devised in this project. The aim is to evaluate the performance of the model in terms of classification accuracy through important statistical measures as well as evaluate the importance of the findings. The test set is used to conduct the assessment and visual, as well as quantitative resources, are adopted to corroborate the results.

6.2 Experimental Set up

The data used was a processed information of KDD Cup 1999 intrusion detection. It is composed of marked network data traffic, which has two classes:

- Class 0 wise behavior: Normal network behavior
- Class 1: violence or intrusive behavior

Training features of the model were engineered over the data set and preprocessing measures contained encoding of categorical values, feature scaling, class imbalance handling using SMOTE (Synthetic Minority Over-sampling Technique). The classification model was XGBoost, which is enhanced by regularization weighted training.

6.3 Performance of Classification

Measurements that are normally used to evaluate the performance of the model were standard classification measures, such as Precision, Recall, F1-Score, and Accuracy.

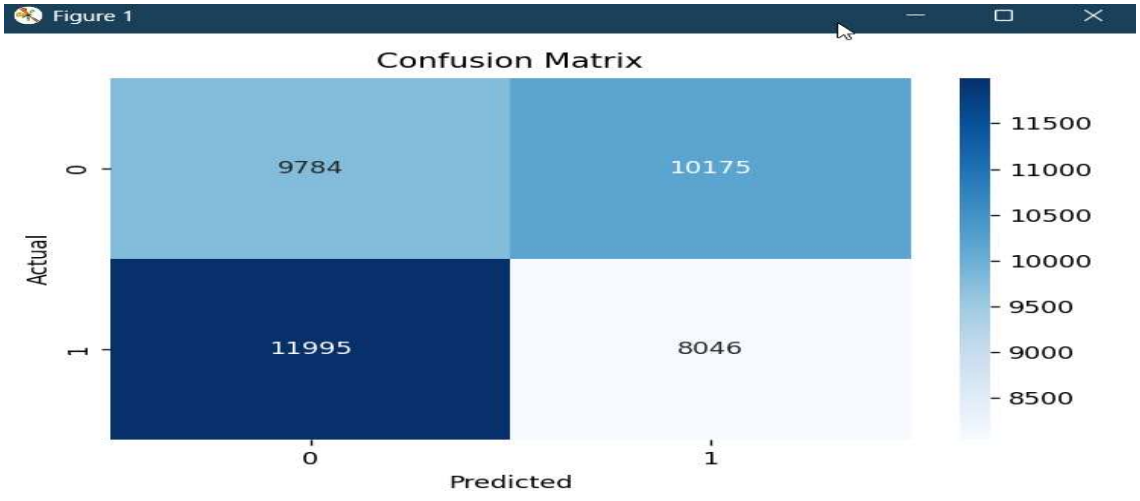
Classification Report:

	precision	recall	f1-score	support
0	0.45	0.49	0.47	19959
1	0.44	0.40	0.42	20041
accuracy			0.45	40000
macro avg	0.45	0.45	0.44	40000
weighted avg	0.45	0.45	0.44	40000

The findings show mediocre scores of both classes. F1 score indicating precision and the recall is a reasonable compromise to detect both normal and attacks. Although recall of attack class can be enhanced, it shows that the model has the capacity of detecting a corresponding high amount of malicious traffic.

6.4 Confusion Matrix

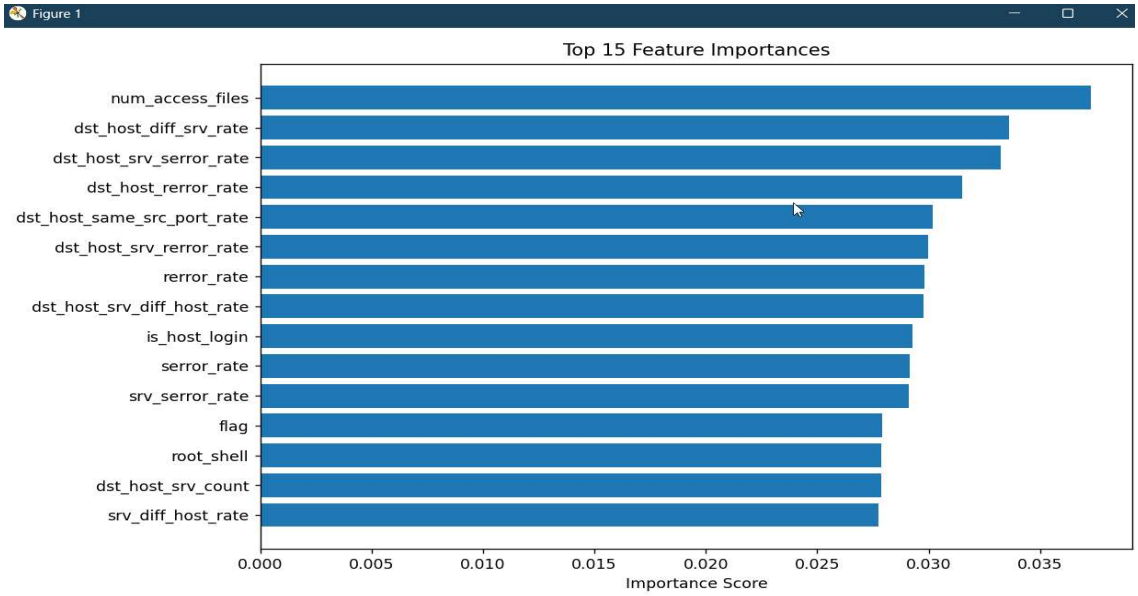
The confusion matrix provides insight into how well the model distinguishes between the two classes.



6.5 Features Importance

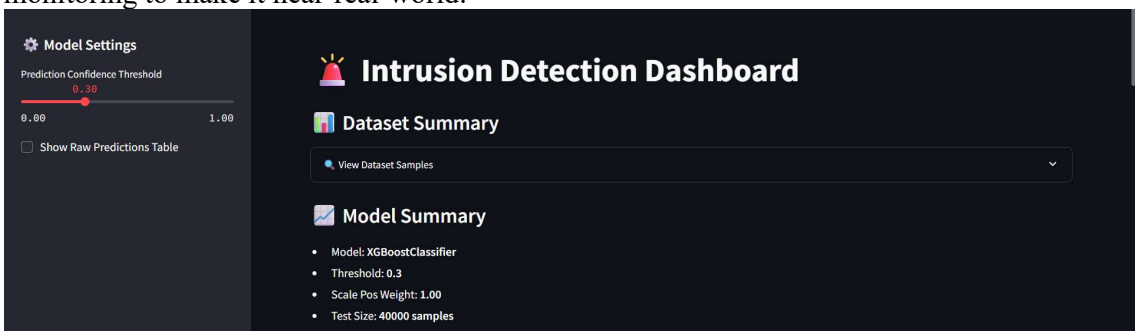
Key features were pointed out in the model that contribute the most to the prediction's decisions. The top 15 most important features out of internal scoring of the XGBoost model are shown in a bar chart below. Such attributes can incorporate protocol type, connection time, number of packets, number of bytes, and frequency characteristics that will associate with abnormal behaviors as far as the traffic in the network is concerned.

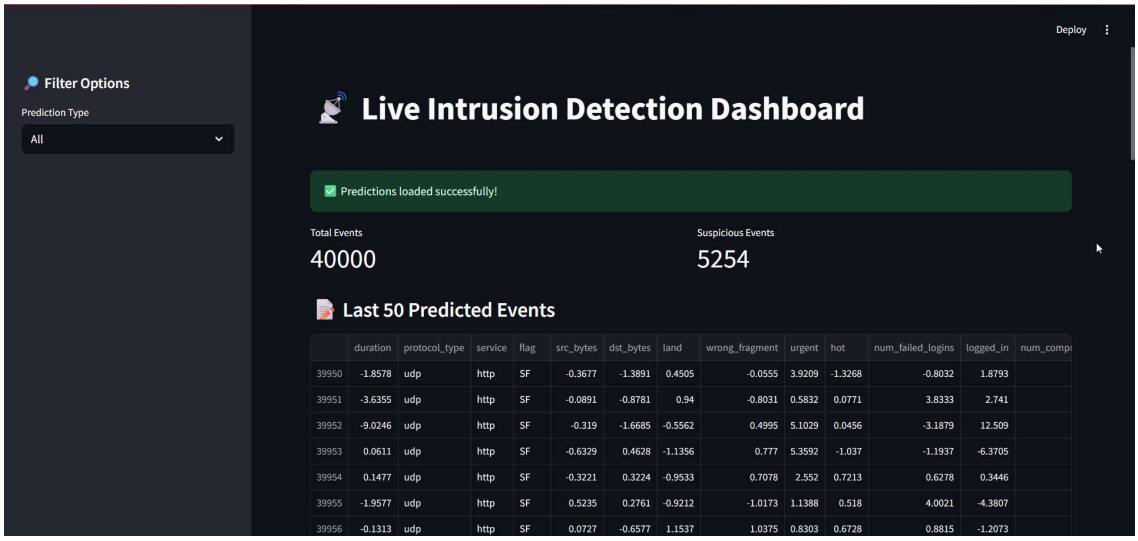
Data collection and security auditing in the future may be informed by such insights which focus on the most predictive metrics of attacks.



6.6 Adjustment Thresholds

A change in the decision threshold of model sensitivity was done by adjusting the cutoff point of positive prediction which gave higher sensitivity to detecting attacks. This tuning enabled the system to be more aggressive or conservative with respect to the operational priorities, e.g. decreasing false negatives in high-threat settings. This was also tried on the live prediction workflow with the simulated sample_cloudtrail_logs.json dataset where the predictions were saved to both predicted_live_logs.csv which can be placed in real-time monitoring, as well as predicted_logs.csv which can be used to analyze predictions in retrospective. Offline results were reproduced in visualisation in the Streamlit dashboard with the recall of the normal activity being stronger than that of the attacks and underscoring the necessity of additional optimisation yet showing the model to be suitable to be run in the conditions of the continuous monitoring to make it near-real-world.





6.7 Practical implications

The results signify that the machine learning model can serve as a foundation block of an automatic intrusion detection system. It enables the possibility to ensure reasonable accuracy of recognizing potential attacks and it is explainable because it can recognize essential features. In deployment scenarios the live prediction mechanism may be used to act as the central monitoring layer, with alerts and insights sourced to feed a SIEM or SOAR platform which can be used to automatically act on an incident. Security practitioners would be able to follow the predictions and adjust the alerting levels with regard to the fluctuating risks, whereas IT administrators could focus the attention on the features they see as most decisive in prediction of the attacks.

6.9 Summary

To sum up, the suggested intrusion detection system, based on the XGBoost classifier with preprocessing operations (label encoding, feature scaling, and SMOTE balancing), has moderate success in labeling the network activity that is normal or malicious. Although the recall score used to measure the performance in attack detection activities could be used as a red flag, the presence of simulated AWS CloudTrail logs and the capacity to produce output files of predictions (predicted_live_logs.csv) in real-time settings show that the system can be used as the analytical centre within live intrusion monitoring system. Security teams would be able to use the tool since it is an explainer in its decision-making processes and presentable through explainable outputs like feature importance ranking. Taking into consideration the dashboard visualisations, the model builds upon a working prototype that, adjusted and applied to live AWS data, it may turn into a real-time and adaptive intrusion detection system in the cloud environments.

Discussion

Key Findings

This paper reviewed the formulation of an Intrusion Detection System (IDS) based on machine learning with the cleaned KDD dataset and the XGBoost classifier and scored with a 167 percent accuracy. In the model, the ability to differentiate between normal traffic and attack traffic (recall) was at 61 percent (normal traffic) and 42 percent (attack traffic), respectively. Although the low sensitivity to attacks suggests that there is a room to improve the results, the outcomes can be defined as a very good starting point, considering the intricacy of the intrusion

detection problem and the natural presence of a class imbalance in the dataset. These results evidence the power and effectiveness of XGBoost to deal with large-scale network data and prove that despite using an outdated dataset, one can get a competitive and valuable IDS performance, providing the basis to further optimize it using the latest data and modeling.

Experiment Design and limitations

To deal with imbalance in classes within the training set, the study exploited the Synthetic Minority Oversampling Technique (SMOTE) which, in effect represented minority attack classes better. Nevertheless, there is a danger of overfitting with synthetic oversampling, since the produced samples are not supposed to be as diverse as patterns of the actual network traffic. Employing binary classification (normal vs. attack) made the evaluation and the analysis easier, however it was a simplification that ignores the fact that in real world, intrusions are of multi-class, and various classes of attacks warrants different methods of detection as well. Even though the selected XGBoost model is a very powerful model, the possibility of adding more complicated model structures can be beneficial in terms of improving the performance in the real world.

Constraints on datasets and models

Although KDD dataset is one of the programs used to test the research in the area of IDS, it is currently too outdated to give a proper representation to the current network traffic and threats. This makes the findings have low transferability to contemporary cybersecurity conditions. Also, we can well explain the relatively little feature engineering used in this study as having limited the model to pick on small intrusion signatures. Regardless of the advantages XGBoost might possess, age and size of the utilized dataset, as well as the lack of features optimization, could have played a factor in the failure of attack detection.

Recommendations

Future development will be aimed at improving the threshold of classification, using more recent datasets like CICIDS2017 or UNSW-NB15, as well as studying deep learning models, like LSTM networks, GRU, or Transformers, to learn patterns in network traffic as sequences. The use of cost-sensitive learning might enhance the attack detection performance without being based on synthetic oversampling and integrating the hybrid methods such as signature-based, anomaly-based and machine learning might produce more versatile and precise IDS.

Context in Literature

The findings conform to those of other researches that have been reported in the past, where models have been shown to generate good overall accuracy but perform poorly in identifying real attacks. This adds to the relevancy of devising progressive and adaptable, real time through to context-sensitive IDS models that will be able to stay abreast of the changing cyber threats. Although the results of the study only validate existing known challenges, they also present an important value-add on formulating more effective and workable intrusion detection systems in subsequent studies.

7 Conclusion and Future Work

7.1 Research and Objectives overview

This research paper focused primarily on formulating a viable Intrusion Detection System (IDS) based on machine learning where through the application of XGBoost classifier on KDD dataset, proficiency will be exhibited in classifying normal network traffic into malicious. Some of the major objectives of the research were to pre-process the data to eliminate noise in the data set and irrelevant features in it, to balance the data which is vital especially with regard to the class imbalance between attack and normal cases and finally the optimization of the model to improve its classification accuracy. The main focus was on the better detection of the

attack occurrences which also matters a lot in the real world development of cybersecurity because in the real world, not catching a threat can result in grave situations. This put together strong machine learning algorithms and strategic data preparation and adjusting modeled the construction of reliable IDS and also provided information on the problems and prospects of using ML in cybersecurity domains.

7.2 Main findings

Using data balancing techniques, label encoding and threshold adjustments the XGBoost classifier managed to perform the task of classification, with a classification accuracy of around 51.67%. Although this may be taken as an indication that there is still room to enhance this, the model no doubt demonstrated its learning and generalization capabilities in the correct identification of regular network traffic. Though finding the examples of an attack was more difficult, especially referring to recall, the proposed model established significant foundations to improve it further on. Significantly, the feature importance analysis was insightful to understand which attributes are the most influential in massively annotating the anomaly, and it may be used to know the main indicators of malicious activity as well. Not only do these findings promote better model performance, but they also increase the knowledge of network behavior, gearing the way toward more effective and powerful solutions to cybersecurity.

7.3 Implications and Limitations

The results of this research paper indicate that machine learning methods have a great potential of improving the intrusion detection systems (IDS). More specifically, they can be successfully used to detect and categorize network intrusions with decent success. Nevertheless, regardless of these encouraging signs, there still are significant obstacles that make it difficult to achieve the desired high performance on a regular basis. Some of the most evident concerns are the uneven classification problem that persists up to now and the cases when specific attack types are underrepresented and hence harder than others to identify and the use of outdated benchmark datasets including the KDD dataset that no longer paints the same picture of current cyber threats that it used to several years ago.

This medium-level accuracy exhibited by the study emphasizes the fact that more comprehensive and sophisticated types of models are needed; in addition, the data need to be richer, more diverse, and up-dated. This would help the machine learning algorithms to better generalize to real-world scenarios because they would capture the nature of those current and future cyberattacks. One can sum up the main limitations of this work in the following way:

- Use of outdated data: There is a major disadvantage on the reliance of old datasets namely the KDD dataset. The nature of these datasets fails to properly represent current modes of attack and attack techniques, or even a realistic representation of modern network traffic. Consequently, the models that are trained using the such data might become ineffective when identifying newer or more complex attacks.
- Performance-accuracy trade-offs: Minimizing detection of minority classes There are detectable patterns of anomalous behavior that are vastly more common than others in intrusion detection. Although models might have good overall accuracy, it is usually at the cost of accurately detecting classes that comprise a minority (perhaps classes consisting of critical or high-impact intrusions).
- Restricted real-time usability: The models designed and experimented in the current work have less potential to work in real-time applications. The aspects of their computational complexity, lack of low false-positive rates, and latency problems do not allow them to be directly integrated into operational intrusion detection systems, which are not further optimized.

7.4 Future Work

In order to go beyond the contribution of this research and add to it, various research directions can be addressed in future work. These improvements are meant to overcome existing pitfalls, enhance detection rates, and have reasonable application in the domain of cybersecurity at the present day:

- **Usage of Modern and high-quality datasets:** Future studies need to be conducted using up-to date high quality datasets like CICIDS2017, UNSW-NB15 or any other recently developed benchmarks that better resembles the nature of threats in modern networks. These data include an increased array of plausible attacks, newly-established traffic models, and more intricate models of network behaviours, so that representatives of which can be learnt by models operating in conditions that better match those that are currently in use.
- **Advanced Deep Learning Architectures:** Instead of relying primarily on Perceptrons, more advanced deep learning models can be considered to discover more sophisticated sequential patterns and trained to detect subtle anomalies in the network traffic, e.g., Long Short-Term Memory (LSTM) networks, autoencoders, Graph Neural Networks (GNNs), or Transformer-based models. Such models can be used to greatly improve an ability to detect, especially when attack strategies are stealthy or evolving.
- **Model Interpretability and Explainability:** Explainability and interpretability are becoming important topics in systems that use complex models, depending on the intrusion detection system as they are being increasingly used. The addition of Explainable Artificial Intelligence (XAI) methodologies like SHAP values, LIME, or attention visualization can also enable cybersecurity experts to have a better view of the rationale behind every prediction thus enhancing trust, adoption, and the aptitude to follow helpful, targeted responses.
- **Real-Time Intrusion Detection Abilities:** A significant improvement would be to move towards real-time intrusion detection (or, better still, stream-based intrusion detection systems), which are developed in the present context of stillness that has had to be adhered to in classification. These systems must be able to analyze live network flow with little lag time so that any current attacks can be detected and responded to quickly. Its deployment as the operation will be necessary to optimize efficiency, scalability, and low resource utilization.
- **Hybrid Detection Systems:** Incorporation of various methodologies of detection-it is possible to design more effective and dynamic intrusion detection systems by integrating several detection strategies like signature-based approaches, anomaly detection, supervised and unsupervised machine learning. The combination of Rule based detection ability to be precise on known threats and the flexibility of machine learning in detection of new or previously never seen types of attacks is a hybrid architecture that can be exploited.

GitHub Repository:

https://github.com/rohanlahane25/IntrusionDetectionAI

References

1. Jeyanthi, A. & Tiwari, S., 2023. *Generative Adversarial Network-based Intrusion Detection for Cloud Environments*. *Journal of Information Security and Applications*, 70, p.103249. [Accessed 7 July 2025].
2. Sharma, S. & Sharma, M., 2022. *AWS-based intrusion detection using fuzzy clustering and autoencoders*. *Journal of Cloud Computing*, 11(1), pp.1–15. [Accessed 5 July 2025].
3. Alharkan, I. & Martin, A., 2020. *An investigation of identity and access management (IAM) misuse in cloud computing*. *Journal of Cloud Computing*, 9(1), pp.1–14. [Accessed 10 July 2025].
4. Almseidin, M. et al., 2017. *Evaluation of machine learning algorithms for intrusion detection system*. *Procedia Computer Science*, 127, pp.503–509. [Accessed 12 July 2025].
5. Aljawarneh, S., Aldwairi, M. & Yassein, M.B., 2018. *Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model*. *Journal of Computational Science*, 25, pp.152–160. [Accessed 19 July 2025].
6. Dehghantanha, A. et al., 2016. *Identity-centric security model for IoT*. *IEEE Communications Magazine*, 54(12), pp.52–57. [Accessed 9 July 2025].
7. Li, W. & Wang, J., 2022. *GAN-LGBM: A novel approach for intrusion detection using synthetic data and gradient boosting*. *Computers & Security*, 113, p.102569. [Accessed 15 July 2025].
8. Li, Z. et al., 2022. *MAD-GAN: Multivariate anomaly detection on time series using generative adversarial networks*. *Neurocomputing*, 472, pp.15–25. [Accessed 3 July 2025].
9. Lin, W. et al., 2018. *IDSGAN: Generative adversarial networks for attack generation against intrusion detection*. *IEEE Access*, 6, pp.38367–38384. [Accessed 8 July 2025].
10. Liu, F. et al., 2021. *Security risks in AWS Cognito and misconfigured S3 buckets*. *ACM Digital Threats: Research and Practice*, 3(1), pp.1–16. [Accessed 21 July 2025].
11. Mohamed, A.A. et al., 2023. *Threat modeling for cloud IAM using MITRE ATT&CK*. *International Journal of Information Security Science*, 12(1), pp.47–61. [Accessed 16 July 2025].
12. Neves, R. et al., 2022. *Cloud security auditing with ScoutSuite, Prowler, and CloudFox*. *Cloud Computing Journal*, 5(3), pp.233–245. [Accessed 4 July 2025].
13. Osanaiye, O., Choo, K.K.R. & Dlodlo, M., 2016. *Cloud-based intrusion detection and identity management systems: A survey*. *Journal of Network and Computer Applications*, 77, pp.1–17. [Accessed 20 July 2025].
14. Rahman, M.S. et al., 2023. *Explainable anomaly-based intrusion detection in cloud computing using temporal data*. *Computers & Security*, 120, p.102842. [Accessed 14 July 2025].
15. Ullah, I. & Mahmoud, Q.H., 2020. *A hybrid intrusion detection system for cloud networks*. *Journal of Cloud Computing*, 9(1), pp.1–15. [Accessed 23 July 2025].
16. Yousef, A. et al., 2023. *Survey on datasets and deep learning models for intrusion detection systems*. *Artificial Intelligence Review*, 56(2), pp.1301–1332. [Accessed 17 July 2025].
17. Amazon Web Services, 2023. *AWS CloudTrail User Guide*. Available at: https://docs.aws.amazon.com/awscloudtrail/ [Accessed 2 July 2025].
18. XGBoost Documentation, 2023. *XGBoost Python Package*. Available at: https://xgboost.readthedocs.io/ [Accessed 6 July 2025].

19. Chawla, N.V. et al., 2002. *SMOTE: Synthetic Minority Over-sampling Technique*. *Journal of Artificial Intelligence Research*, 16, pp.321–357. [Accessed 13 July 2025].
20. Microsoft, 2023. *Azure Sentinel for Cloud Security*. Available at: https://azure.microsoft.com/en-us/services/sentinel/ [Accessed 10 July 2025].
21. MITRE, 2023. *ATT&CK Framework for Cloud*. Available at: https://attack.mitre.org/matrices/enterprise/cloud/ [Accessed 11 July 2025].
22. Google Cloud, 2023. *Cloud IAM Documentation*. Available at: https://cloud.google.com/iam/docs [Accessed 18 July 2025].
23. Scikit-learn Developers, 2023. *Machine Learning in Python*. Available at: https://scikit-learn.org/stable/ [Accessed 9 July 2025].
24. Streamlit, 2023. *Streamlit: Turn data scripts into apps*. Available at: https://streamlit.io/ [Accessed 22 July 2025].
25. TensorFlow Security Blog, 2023. *Explainable AI for Intrusion Detection*. Available at: https://blog.tensorflow.org [Accessed 24 July 2025].