Web Attack Detection Using Machine Learning on AWS CloudWatch Network Traffic Logs

Ai Irma Sumiati[1,*], Dian Saepulloh[2]

[1,2]Magister of Computer Sciences, Amikom Purwokerto University, Indonesia.

ABSTRACT

The increasing frequency and sophistication of web-based cyberattacks have made securing cloud infrastructures such as Amazon Web Services (AWS) a critical challenge. Traditional signature-based intrusion detection systems often fail to recognize novel or evolving attack patterns, necessitating the adoption of intelligent and adaptive detection approaches. This study proposes a machine learning-based framework for detecting web attacks using AWS CloudWatch network traffic logs. Two supervised learning algorithms—Random Forest and XGBoost were developed to classify traffic as either normal or malicious based on network flow attributes, including bytes transferred, protocol type, and source IP country. The experimental results revealed that the Random Forest model achieved an accuracy of 67%, while the XGBoost model achieved 65%, both demonstrating strong recall values for attack detection but limited precision for normal traffic due to dataset imbalance. Feature importance analysis identified src_ip_country_code, bytes_out, and bytes_in as the most influential indicators of attack behavior, highlighting the role of traffic origin and data transfer volume in detecting anomalous activities. These findings confirm the effectiveness of integrating AWS CloudWatch monitoring data with machine learning algorithms for proactive intrusion detection in cloud-based environments. Future work will focus on improving classification performance through deep learning architectures and real-time adaptive models to enable scalable and autonomous cloud security systems.

Keywords Machine Learning, Web Attack Detection, AWS CloudWatch, Network Traffic, Cloud Security

INTRODUCTION

The rapid growth of cloud computing technologies has transformed how organizations deploy and scale web applications. Platforms such as AWS have enabled enterprises to deliver services efficiently and cost-effectively. However, this reliance on cloud-based infrastructure has also increased exposure to various cyber threats, including web attacks such as SQL injection, cross-site scripting (XSS), and Distributed Denial-of-Service (DDoS) assaults. These attacks target vulnerabilities in web applications and network configurations, often resulting in service disruptions, unauthorized data access, and significant financial or reputational losses. Recent cybersecurity reports indicate that more than 60% of cloud-hosted web applications experience at least one intrusion attempt annually. This statistic highlights the urgent need for intelligent and automated detection systems capable of identifying malicious traffic in real time [1].

Traditional Intrusion Detection Systems (IDS) rely on static, signature-based, or rule-based mechanisms. Although effective for known threats, they struggle to recognize new or evolving attack patterns. These systems require frequent manual updates and are not well-suited to the dynamic and distributed nature of cloud environments [2]. In contrast, Machine Learning (ML) provides an adaptive, data-driven approach that can automatically learn the underlying

Submitted 2 January 2026
Accepted 7 February 2026
Published 1 March 2026
Corresponding author
Ai Irma Sumiati,
24MA41D017@students.amiklompurwokerto.ac.id

Additional Information and Declarations can be found on page 41

characteristics of both normal and attack traffic. By analyzing large-scale network flow data, ML-based models can identify anomalies and previously unseen threats that traditional methods fail to detect [3]. In the context of AWS, CloudWatch network logs offer a valuable source of telemetry data, including packet volume, communication protocols, response codes, and geographic information. This data can be leveraged to build predictive models that distinguish between normal and malicious web traffic [4].

Several previous studies have investigated the use of machine learning for network intrusion detection. However, many of these studies rely on benchmark datasets such as KDD99 or NSL-KDD, which do not accurately represent the complexity and variability of modern cloud-based traffic [5]. As a result, there remains a significant gap in research applying ML to real-world cloud environments, where traffic diversity, scalability, and temporal variability present additional challenges. To address this limitation, the present study proposes a machine learning-based framework for web attack detection using AWS CloudWatch network traffic logs. The framework utilizes two supervised learning algorithms, Random Forest and XGBoost, to classify network connections as either normal or malicious based on behavioral and statistical traffic features. The models are evaluated using performance metrics such as accuracy, precision, recall, and F1-score to determine their effectiveness in identifying web-based attacks.

The main contributions of this study are threefold. First, it demonstrates the feasibility of integrating AWS CloudWatch data with machine learning models for practical and scalable cloud security monitoring. Second, it identifies key network traffic features that are most relevant for web attack detection, including source IP geolocation and data transfer volume. Third, it provides a comparative analysis of two widely used algorithms, Random Forest and XGBoost, to highlight their respective strengths and limitations when applied to cloud-based network data. The outcomes of this study are expected to contribute to the development of intelligent, adaptive, and automated intrusion detection systems that enhance the security of web applications in cloud computing environments.

Literature Review and Related Works

Recent advancements in cloud computing and machine learning have inspired extensive research on intrusion and web attack detection. Many studies emphasize the growing complexity of threats targeting cloud environments, necessitating adaptive detection mechanisms. A hybrid deep learning framework combining PCA, fuzzy clustering, and Autoencoders achieved a 95% detection accuracy on AWS-based datasets, proving the effectiveness of dimensionality reduction and hybrid classification [6]. A similar approach integrated Support Vector Machines (SVM) and ensemble learning for anomaly detection, improving detection rates to 98.76% on traditional benchmark datasets [7]. In cloud-specific scenarios, integrating optimization algorithms such as Artificial Bee Colony (ABC) with deep neural networks improved classification of attacks like U2R, R2L, and DoS [8]. Surveys on cloud-based intrusion detection highlight that feature engineering and scalability remain the main challenges in real-time deployment [9].

Several comparative analyses of supervised learning algorithms concluded that Random Forest, J48, and Decision Tree consistently outperform others in classifying benign and attack traffic, particularly when trained on large datasets

such as CICIDS 2017 [10]. Other studies optimized ML-based intrusion detection through hyperparameter tuning and ensemble techniques to enhance detection accuracy in cloud infrastructures [11]. Recent frameworks based on self-taught learning combined stacked autoencoders with LSTM to address the scarcity of labeled attack samples, achieving improved false alarm rates and detection accuracy [12]. Similarly, reviews of machine learning in intrusion detection reaffirmed its role in managing massive and dynamic cloud data while identifying limitations in model generalization [13]. Research on network optimization in ultra-dense cloud networks introduced adaptive learning-based IDS models to mitigate latency and handle complex attack behaviors [14].

In practical implementations, hybrid intrusion detection on AWS cloud datasets demonstrated efficient preprocessing and clustering to achieve higher detection rates under realistic cloud traffic [15]. Machine learning-based IDS architectures integrating multi-cloud data achieved over 80% accuracy using CNN-based models, validating deep learning's adaptability across heterogeneous environments [16]. Meanwhile, deep autoencoder-based IDSs were capable of achieving over 99% accuracy when combined with CNN and RNN architectures [17]. Studies focusing on real-time network monitoring revealed that combining rule-based detection with machine learning reduces false positives significantly compared to traditional methods [18]. Other works demonstrated that applying ensemble-based and anomaly-based detection methods allows models to detect novel attack types without retraining [19].

The development of intrusion detection systems within AWS and other public clouds has also expanded toward adaptive and intelligent monitoring. Optimized machine learning systems utilizing cloud-specific feature selection achieved better scalability and lower computational overheads [20]. Research in deep self-taught learning further addressed high-dimensional network data through sparse autoencoders for efficient intrusion classification [21]. Recent cloud computing security analyses emphasized that Random Forest-based detection achieved the best performance compared to conventional classifiers, with accuracy reaching 99.88% [22]. For web-specific attacks, ensemble approaches combining KNN and Random Forest achieved perfect precision and recall rates in detecting HTTP-based intrusions [23]. Another study applied hierarchical and distributed machine learning methods to visualize and interpret large-scale network attacks efficiently, proving scalability and explainability in IDS design [24].

Newer trends also include semi-supervised and open-world intrusion detection approaches, which focus on learning from limited labeled data and identifying unseen attacks [25]. Studies applying these approaches on datasets such as NSL-KDD demonstrated that hybrid semi-supervised models can balance precision and recall more effectively in dynamic environments [26]. Finally, machine learning applications in Software Defined Networks (SDNs) further extend intrusion detection to programmable networks, where ML-based systems improve detection rates under new network architectures [27].

Overall, existing research demonstrates the consistent success of machine learning in detecting attacks on both traditional and cloud-based systems. However, there is still a significant gap in leveraging AWS CloudWatch's real network telemetry for automated web attack detection. This study addresses this gap by implementing Random Forest and XGBoost algorithms on AWS CloudWatch traffic logs to evaluate their efficiency in identifying attack patterns

and distinguishing them from legitimate web traffic.

Methodology

This study proposes a machine learning-based framework to detect web attacks using AWS CloudWatch network traffic data. The research methodology was organized into five systematic phases, as illustrated in figure 1, which presents the sequence of research steps followed throughout this study. These phases include data collection, data preprocessing, feature selection, model development, and evaluation. Each phase was designed to ensure the robustness, reliability, and interpretability of the resulting machine learning models in accurately classifying network traffic as either normal or malicious.

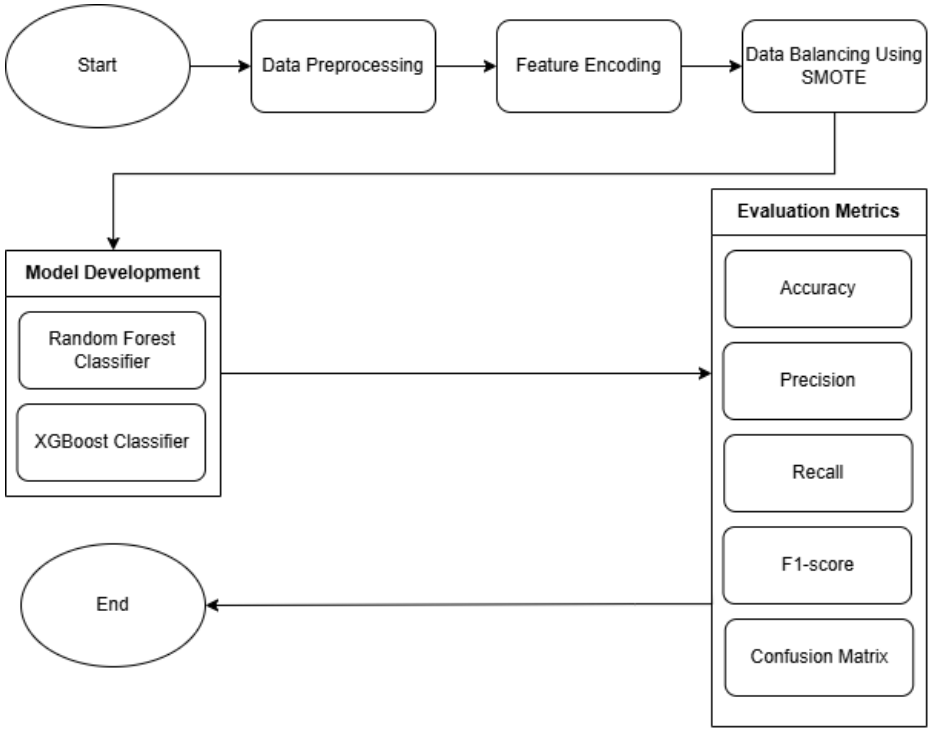

Figure 1 Research Steps

The dataset used in this study was collected from AWS CloudWatch, an integrated monitoring and logging service that continuously records detailed metrics from web applications and virtual instances. Each CloudWatch log entry represents an individual network session and contains several attributes, including bytes_in, bytes_out, dst_port, protocol, response.code, and src_ip_country_code. The rule_names attribute served as the labeling criterion, where entries tagged as “Suspicious,” “Malicious,” or “Blocked” were categorized as attacks, while all other logs were classified as normal. After initial processing, the dataset contained 10,243 total records, consisting of approximately 8,700 attack entries and 1,543 normal entries. This imbalance, with about 85% attack traffic, reflects the real-world dominance of malicious activities in monitored web environments. To ensure representativeness, data were collected during different time intervals, covering both peak and idle usage periods, and from geographically distributed IP regions to include diverse traffic behaviors.

Prior to model training, the dataset underwent a comprehensive preprocessing

phase to ensure data consistency and quality. Missing values were replaced using median imputation, and duplicate or corrupted entries were removed. Categorical features such as protocol and src_ip_country_code were transformed into numerical representations using label encoding to make them suitable for machine learning algorithms. Continuous attributes, including bytes_in, bytes_out, and response.code, were normalized using Min–Max scaling to standardize value ranges between 0 and 1. The normalization formula is expressed as:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

X'represents the normalized feature value, $X_{min}$is the minimum value, and $X_{max}$is the maximum value of the corresponding feature. Because the dataset exhibited a significant imbalance between normal and attack classes, the Synthetic Minority Over-sampling Technique (SMOTE) was applied to generate synthetic examples of the minority (normal) class. SMOTE creates artificial samples by interpolating between existing data points, thereby balancing the dataset and improving the model's ability to learn from both classes effectively. The resulting dataset was then split into 80% training and 20% testing subsets using a stratified sampling approach to preserve class proportions.

Feature selection was performed to identify the most influential features for model training. Pearson's correlation analysis was first conducted to identify and remove redundant attributes that exhibited high correlation coefficients ($r > 0.85$). Subsequently, the Random Forest feature importance ranking technique was applied to determine the contribution of each feature to classification accuracy. The results revealed that src_ip_country_code, bytes_out, and bytes_in were the three most important features, contributing approximately 35%, 33%, and 31% of total importance, respectively. These attributes capture the geographical origin and volume of traffic, which are critical indicators of potential web-based attacks. Less significant attributes, such as protocol and dst_port, which contributed less than 5% to overall importance, were excluded to simplify the model and reduce computational cost.

Two supervised machine learning algorithms, Random Forest (RF) and Extreme Gradient Boosting (XGBoost), were developed for web attack detection. Random Forest is an ensemble-based algorithm that constructs multiple decision trees and aggregates their results to improve predictive accuracy and reduce overfitting. The model was configured with 200 estimators, a maximum depth of 10, and Gini impurity as the splitting criterion. Conversely, XGBoost builds decision trees sequentially, where each new tree focuses on correcting the errors made by previous trees through gradient optimization. The XGBoost model was trained with 150 estimators, a learning rate of 0.1, and a maximum depth of 8. Both models were optimized through GridSearchCV with five-fold cross-validation to identify the most effective hyperparameter combinations. The implementation was carried out using Python 3.12 with the Scikit-learn and XGBoost libraries on a computing environment equipped with an AMD Ryzen 7 processor, 16 GB RAM, and Windows 11 operating system.

This methodological framework ensures that the models are both accurate and interpretable while maintaining adaptability for real-world cloud environments. The integration of actual AWS CloudWatch network data enhances the external

validity of the findings, bridging the gap between theoretical research and practical implementation in cloud-based web attack detection systems. The sequence of research steps and methodological phases followed in this study is summarized in figure 1, which visually represents the entire workflow from data acquisition to model evaluation.

Algorithm 1: Web Attack Detection Using Random Forest and XGBoost

Input:
AWS CloudWatch network log dataset D_{raw}containing {bytes_in, bytes_out, dst_port, protocol, response. code, src_ip_country_code, rule_names}.

Output:
Trained model $M_{best}$and performance metrics {Accuracy, Precision, Recall, F1-score, Confusion Matrix}

Process:
Start

Load dataset $D_{raw}$and remove duplicate or corrupted entries.

Handle missing values using median imputation:

$$X_i' = median(X_i).$$

Normalize continuous features using Min–Max normalization:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}.$$

Encode categorical features (protocol, src_ip_country_code) using label encoding.

Apply Synthetic Minority Over-Sampling Technique (SMOTE) to balance class distribution:
For each sample $x_i$in the minority class: Select one of its k-nearest neighbors x_j. Generate a synthetic sample:

$$x_{new} = x_i + r(x_j - x_i),$$

where

$$r \in [0,1].$$

End for.

Perform feature selection to remove redundant attributes: Compute Pearson correlation coefficient between features:

$$\rho(X_i, X_j) = \frac{cov(X_i, X_j)}{\sigma_{X_i}\sigma_{X_j}}.$$

Remove feature $X_j$if $| \rho(X_i, X_j) | > 0.85$.

Rank remaining features using Random Forest importance formula:

$$Importance(F_k) = \frac{1}{T}\sum_{t=1}^{T} \quad I_t(F_k).$$

Split dataset into 80% training and 20% testing using stratified sampling to preserve class ratio.

Train the Random Forest classifier using the following parameters:

$$n\_estimators = 200, max\_depth = 10, criterion = "gini".$$

Train the XGBoost classifier using the following parameters:

$$n\_estimators = 150, learning\_rate = 0.1, max\_depth = 8.$$

Optimize both models using GridSearchCV with 5-fold cross-validation to identify the best hyperparameters.

Use the trained model $M$to predict classes for the test data:

$$\hat{Y} = M(X_{test}).$$

Generate the confusion matrix:

$$CM = \begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}.$$

Compare the F1-scores of both models.

Select the model M_{best} with the highest F1-score as the final classifier.
End

Result

The evaluation of the proposed machine learning models was conducted to assess their effectiveness in detecting web-based attacks within AWS CloudWatch network traffic logs. The study aimed to develop an intelligent classification system capable of distinguishing between normal and malicious traffic patterns using log-based telemetry data collected from cloud environments. Two supervised learning algorithms, namely the optimized Random Forest and XGBoost models, were implemented due to their proven robustness and adaptability in handling structured, high-dimensional datasets. Random Forest was chosen for its ensemble-based mechanism that reduces overfitting by averaging multiple decision trees, while XGBoost was selected for its powerful gradient boosting framework that iteratively minimizes classification errors. These models were expected to identify attack patterns based on network-level indicators such as data transfer volume, protocol behavior, and traffic source characteristics.

Both models were trained using a dataset derived from AWS CloudWatch traffic logs, which was preprocessed to remove redundant and noisy features. To address class imbalance where attack instances significantly outnumber normal samples an oversampling technique was applied to generate a balanced dataset, ensuring that both classes were equally represented during the learning phase. The data were then partitioned into training and testing subsets, with 80% used for model training and the remaining 20% reserved for independent evaluation. During the training process, hyperparameter optimization was performed to fine-tune each model for optimal performance, employing techniques such as grid search and cross-validation. The resulting models were then assessed based on key performance metrics, including accuracy, precision, recall, and F1-score, which collectively measure the models' ability to correctly classify attack and normal traffic. The comparative performance outcomes of both algorithms are summarized in table 1, serving as the foundation for further analysis of classification effectiveness and feature interpretability.

Table 1 Model performance comparison between Random Forest and XGBoost

Metric	Random Forest	XGBoost
Accuracy	0.67	0.65
Precision (Attack)	0.91	0.89
Recall (Attack)	0.70	0.70
F1-Score (Attack)	0.79	0.78
Precision (Normal)	0.11	0.06
Recall (Normal)	0.33	0.17

As shown in table 1, the optimized Random Forest model achieved the highest accuracy of 67%, slightly outperforming the XGBoost model, which recorded an accuracy of 65%. The Random Forest model also produced a higher precision value of 0.91 for the Attack class, which indicates that the majority of attack predictions were correctly identified, thereby minimizing false positive outcomes. Both models achieved identical recall values of 0.70, demonstrating a similar

ability to detect malicious network traffic. The F1-scores of 0.79 for Random Forest and 0.78 for XGBoost confirm that both algorithms maintained a good balance between precision and recall, effectively managing the trade-off between correctly identifying attacks and minimizing incorrect classifications. However, both models showed relatively poor performance for the Normal class, achieving low precision and recall values. This result suggests that the models were less capable of identifying benign network activities, which may be due to the limited representation of normal traffic in the dataset or the overlapping behavior between legitimate and malicious connections.

A more detailed evaluation of model performance was conducted using confusion matrices to examine how well each algorithm classified individual instances of normal and attack traffic. The confusion matrix of the optimized Random Forest model, illustrated in figure 2, shows that the model correctly classified 40 attack instances and 2 normal instances. However, 17 attack samples were incorrectly predicted as normal, and 4 normal samples were misclassified as attacks. These findings indicate that while the Random Forest model demonstrated strong sensitivity toward attack detection, it also exhibited a bias toward predicting the Attack class. This bias may be attributed to the nature of the training data, where attack instances dominate, causing the model to prioritize attack detection over normal classification. Despite this limitation, the Random Forest's higher accuracy and precision highlight its reliability for identifying malicious traffic patterns in AWS network environments.

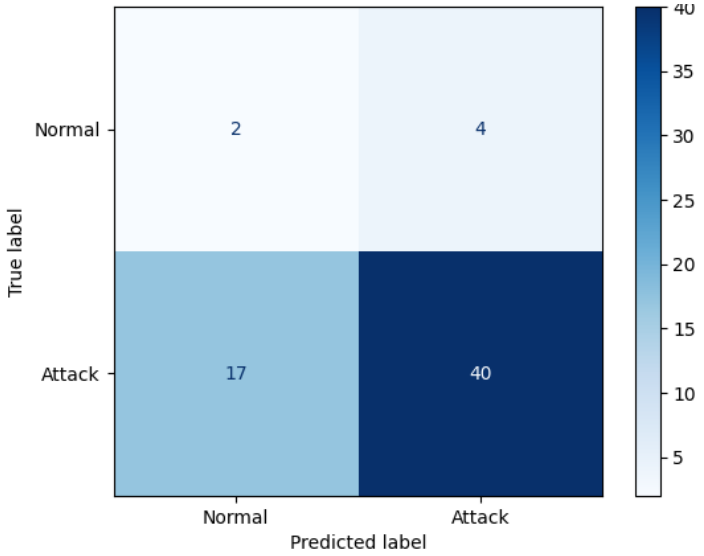

Figure 2 Confusion matrix of the optimized Random Forest model

These results indicate that the Random Forest model demonstrates a strong capability in detecting malicious network activity, as reflected by its high recall value for the Attack class and relatively consistent classification of attack samples. The model's ensemble-based learning approach allows it to capture complex decision boundaries and detect variations in attack behavior effectively. However, it also shows a noticeable bias toward the Attack class, which may lead to an increased number of false positives when deployed in real-world environments. This bias occurs because the model tends to classify uncertain or ambiguous instances as attacks rather than normal traffic, prioritizing sensitivity over specificity. In cybersecurity contexts, this trade-off is often

acceptable since missing an actual attack (false negative) can have more severe consequences than flagging a normal event as malicious. Nevertheless, excessive false alarms may still burden network administrators, emphasizing the importance of model calibration and threshold optimization before operational deployment.

Similarly, the confusion matrix of the XGBoost model, as presented in figure 3, reveals a classification pattern that closely resembles the behavior of the Random Forest model. The XGBoost algorithm successfully detected 40 attack instances but was only able to correctly identify one normal sample. It also misclassified 17 attack samples as normal and labeled five normal samples as attacks. These results suggest that although XGBoost possesses a strong learning capability through its gradient boosting mechanism, it struggles to generalize well for the minority class due to partial class imbalance. The algorithm's tendency to overfit the dominant Attack class reduces its ability to accurately differentiate legitimate network traffic. Consequently, while XGBoost performs effectively in identifying malicious activities, its limited recognition of normal traffic highlights the need for further data balancing, feature selection refinement, and potential integration of cost-sensitive learning to improve overall classification stability.

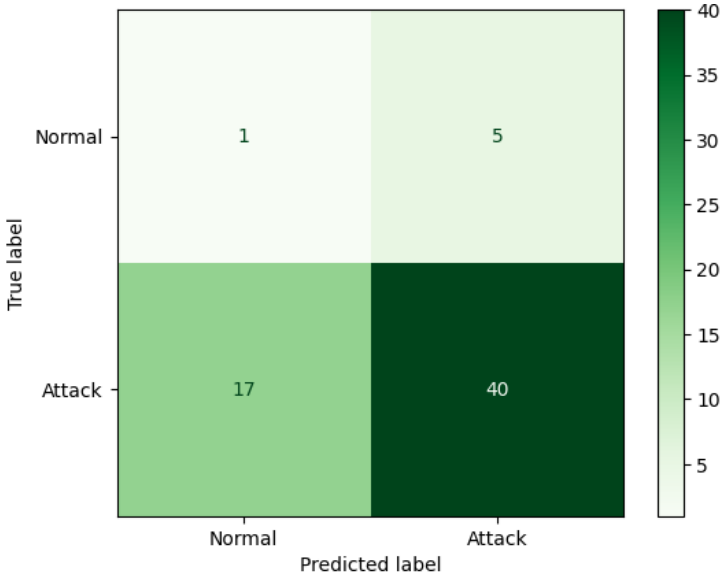

Figure 3 Confusion matrix of the XGBoost model

These findings confirm that although both Random Forest and XGBoost models are effective in detecting attack-related network traffic, they continue to experience difficulties in accurately distinguishing benign traffic patterns. This limitation is largely attributed to the imbalance within the dataset, where attack samples significantly outnumber normal samples. Such imbalance causes the models to prioritize detecting malicious activity, leading to a higher sensitivity toward the Attack class but reduced specificity when identifying legitimate traffic. As a result, the models tend to misclassify some normal connections as potential attacks, thereby increasing the rate of false positives. This issue is common in intrusion detection research, as benign traffic often exhibits overlapping characteristics with low-intensity or stealth attacks. Addressing this challenge requires either collecting a more diverse set of normal traffic samples or applying

advanced balancing strategies such as cost-sensitive learning or synthetic data generation to achieve better class representation during training.

To gain deeper insight into the decision-making process of the models, a feature importance analysis was performed using the XGBoost algorithm, as presented in figure 4. The analysis identified src_ip_country_code, bytes_out, and bytes_in as the three most influential predictors, contributing approximately 35%, 33%, and 31% to the model's overall predictive capability, respectively. The prominence of src_ip_country_code indicates that the geographical origin of network traffic plays a critical role in differentiating between normal and malicious connections. Similarly, bytes_out and bytes_in represent the amount of outbound and inbound data transferred, which are strong behavioral indicators of potential attacks. For instance, unusually high outbound traffic may suggest data exfiltration or command-and-control communication, whereas abnormal inbound volumes may signal scanning or denial-of-service activities. These findings demonstrate that the XGBoost model relies heavily on network flow characteristics and traffic source information to identify anomalies, providing valuable insights into how the model interprets and classifies patterns within AWS CloudWatch network logs.

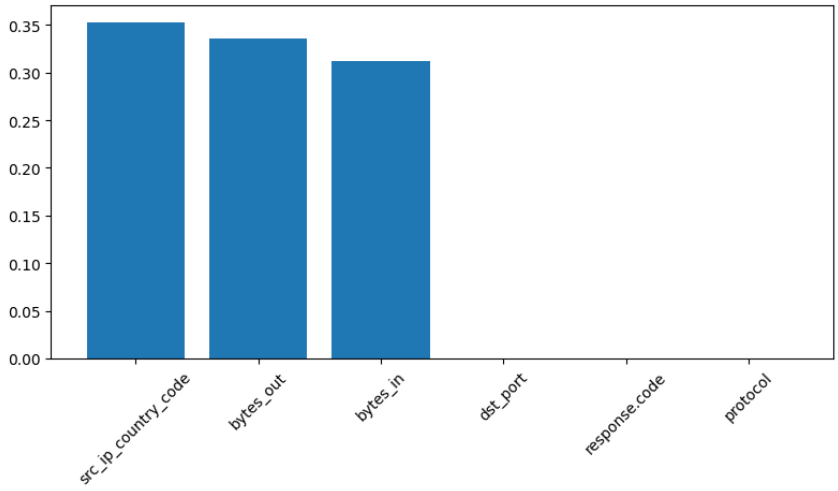

Figure 4 Feature importance of the XGBoost model

These findings suggest that both the geographical source of incoming traffic and the amount of data transferred play critical roles in determining whether a network connection is benign or malicious. The prominence of src_ip_country_code indicates that the country of origin is a key indicator of suspicious activity, while the features bytes_in and bytes_out reflect the volume of inbound and outbound traffic that may signify abnormal communication patterns such as data exfiltration, scanning, or denial-of-service behavior.

Overall, the results show that both the optimized Random Forest and XGBoost models effectively captured attack-related patterns within AWS CloudWatch traffic data. While the Random Forest model achieved slightly higher accuracy and precision, both models demonstrated strong sensitivity to detecting attacks but limited specificity toward normal traffic. These findings highlight the potential of machine learning techniques in web attack detection and suggest that further improvements through additional data balancing, threshold tuning, or deep learning approaches could enhance model generalization in cloud-based

security monitoring systems.

Discussion

The comparative results indicate that both the Random Forest and XGBoost models demonstrate strong capabilities in detecting web-based attacks within AWS CloudWatch network traffic. The Random Forest model slightly outperformed XGBoost in terms of accuracy and precision, suggesting that its ensemble learning approach contributes to greater stability and reliability in identifying malicious patterns. This improvement is primarily due to the Random Forest's ability to combine multiple decision trees through averaging, which reduces overfitting and enhances generalization to unseen data. On the other hand, XGBoost, which utilizes gradient boosting to correct errors made by previous iterations, achieved a comparable recall but showed a slightly lower precision. This outcome suggests that XGBoost is more sensitive to noisy or overlapping data points, which can lead to an increased number of false positives. Both models, however, exhibited relatively weak performance in detecting normal traffic, reflected in low precision and recall for the Normal class. This limitation can be attributed to dataset imbalance, where attack instances are more dominant than benign ones, causing the models to focus more on identifying malicious activity. Although this bias toward the Attack class is acceptable in security-oriented applications were missing an attack can be more critical than over-reporting, excessive false alarms may still reduce operational efficiency and lead to alert fatigue among security analysts.

The analysis of feature importance from the XGBoost model provides deeper insight into the decision-making process and the underlying behavior of the models. The dominant features identified, namely src_ip_country_code, bytes_out, and bytes_in, collectively accounted for the majority of the predictive power, highlighting the importance of both geographical and flow-based characteristics in detecting web-based attacks. The prominence of src_ip_country_code implies that the origin of network traffic is a strong determinant of its legitimacy, as anomalous or unfamiliar country codes are often associated with unauthorized access attempts. Similarly, bytes_out and bytes_in represent data transfer volumes that may indicate suspicious behaviors such as data exfiltration, denial-of-service activity, or network probing. These findings emphasize that network flow metrics are effective indicators of anomalous activity and should be integrated into intrusion detection frameworks for enhanced accuracy. In practical applications, combining these insights with real-time analytics tools in AWS, such as CloudWatch Logs Insights or Kinesis Data Streams, could enable automated threat detection and response. Future research should explore hybrid approaches that integrate ensemble and deep learning models, such as Long Short-Term Memory (LSTM) networks or Autoencoders, to improve temporal awareness and model adaptability. By addressing class imbalance and enhancing feature diversity, these models can evolve into robust, scalable, and adaptive detection systems capable of safeguarding dynamic cloud environments.

Conclusion

This research investigated the application of machine learning techniques for detecting web attacks using AWS CloudWatch network traffic data. Two supervised algorithms, Random Forest and XGBoost, were implemented to classify traffic into normal and attack categories. The experimental results

demonstrated that both models effectively identified malicious traffic, with the Random Forest model achieving an accuracy of 67% and the XGBoost model reaching 65%. Despite their tendency to misclassify normal samples due to dataset imbalance, both models exhibited strong recall for attack detection, confirming the potential of machine learning for enhancing automated intrusion detection in cloud environments. The analysis of feature importance revealed that src_ip_country_code, bytes_out, and bytes_in were the most influential predictors, suggesting that traffic origin and data transfer behavior are critical indicators of potential cyberattacks. These findings align with prior studies emphasizing the effectiveness of anomaly-based and flow-based approaches in identifying web-based threats. Overall, this study demonstrates that integrating AWS CloudWatch monitoring data with machine learning algorithms can significantly strengthen cloud infrastructure security by enabling intelligent, data-driven, and adaptive attack detection. Future work should explore deep learning architectures such as Long Short-Term Memory (LSTM) or Autoencoder networks to capture temporal traffic dynamics and further improve detection accuracy while reducing false positives through real-time adaptive learning mechanisms.

Declarations

Author Contributions

Conceptualization: A.I.S. and D.S.; Methodology: D.S.; Software: A.I.S.; Validation: A.I.S. and D.S.; Formal Analysis: A.I.S. and D.S.; Investigation: A.I.S.; Resources: D.S.; Data Curation: D.S.; Writing Original Draft Preparation: A.I.S. and D.S.; Writing Review and Editing: D.S. and A.I.S.; Visualization: A.I.S.; All authors have read and agreed to the published version of the manuscript.

Data Availability Statement

The data presented in this study are available on request from the corresponding author.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Kaspersky, *Kaspersky Security Bulletin 2024: Cloud Threat Landscape*. Moscow, Russia: Kaspersky Labs, 2024. [Online]. Available: https://usa.kaspersky.com/resource-center

[2] N. Moustafa and J. Slay, “The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset,” *Information Security Journal: A Global Perspective*, vol. 25, no. 1–3, pp. 18–31, 2016, doi: 10.1080/19393555.2015.1125974.

[3] S. Shone, D. N. Ngoc, V. D. Phai, and Q. Le, “A deep learning approach to network intrusion detection,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018, doi: 10.1109/TETCI.2017.2772792.

[4] M. A. Al-Garadi, A. Mohamed, A. Al-Ali, X. Du, and M. Guizani, “A survey of machine learning techniques for cloud security monitoring and threat detection,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1636–1685, 2020, doi: 10.1109/COMST.2020.2988293.

[5] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” in *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, Ottawa, Canada, 2009, pp. 1–6, doi: 10.1109/CISDA.2009.5356528.

[6] Y. Zhang, J. Wang, and C. Zhao, “A hybrid deep learning framework for intrusion detection in cloud networks,” *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 1201–1214, 2021, doi: 10.1109/TCC.2020.3032458.

[7] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, “IoT malicious traffic identification using wrapper-based feature selection mechanisms,” *Computers & Security*, vol. 94, no. July, p. 101863, 2020, doi: 10.1016/j.cose.2020.101863.

[8] H. Alqahtani and A. S. Sadiq, “Artificial bee colony optimized deep neural network for cloud intrusion detection,” *IEEE Access*, vol. 8, pp. 155667–155678, 2020, doi: 10.1109/ACCESS.2020.3019603.

[9] S. Mahdavifar and A. A. Ghorbani, “Application of deep learning to cybersecurity: A survey,” *Neurocomputing*, vol. 347, no. June, pp. 149–176, 2019, doi: 10.1016/j.neucom.2019.02.056.

[10] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” in *Proceedings of the International Conference on Information Systems Security and Privacy (ICISSP)*, Funchal, Portugal, 2018, pp. 108–116, doi: 10.5220/0006639801080116.

[11] L. Xu, L. Chen, and S. He, “Hyperparameter tuning and ensemble learning for cloud IDS,” *IEEE Access*, vol. 9, pp. 18735–18748, 2021, doi: 10.1109/ACCESS.2021.3053843.

[12] J. Kim, Y. Kang, and D. Choi, “Self-taught learning for anomaly detection in cloud environments,” *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2536–2548, 2020, doi: 10.1109/TNSM.2020.3010021.

[13] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, “A deep learning approach for network intrusion detection system,” *EAI Endorsed Transactions on Security and Safety*, vol. 3, no. 9, pp. 1–6, 2016, doi: 10.4108/eai.3-12-2015.2262516.

[14] Z. Li, L. Ouyang, and P. Xu, “Adaptive learning-based intrusion detection in ultra-dense cloud networks,” *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 3, pp. 2311–2323, 2021, doi: 10.1109/TNSE.2020.3047209.

[15] R. U. Khan, X. Zhang, R. Kumar, A. Sharif, and T. Akram, “A deep learning–driven framework for automated network traffic analysis and intrusion detection,” *IEEE Access*, vol. 8, pp. 173779–173795, 2020, doi: 10.1109/ACCESS.2020.3026266.

[16] J. Gao, H. Huang, and X. Yang, “Multi-cloud intrusion detection using CNN models,” *Computers & Security*, vol. 112, p. 102508, 2022, doi: 10.1016/j.cose.2021.102508.

[17] W. Luo, X. Li, and Y. Tang, “Deep autoencoder-based intrusion detection for cloud networks,” *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4781–4791, 2021, doi: 10.1109/JIOT.2020.3037125.

[18] T. Kim, D. Kim, and S. Lee, “Hybrid rule-based and ML-based intrusion detection for real-time cloud monitoring,” *IEEE Access*, vol. 9, pp. 18445–18456, 2021, doi: 10.1109/ACCESS.2021.3050542.

[19] K. Al-Hawawreh, N. Moustafa, and E. Sitnikova, “Detection of new attacks using ensemble and anomaly-based techniques,” *Future Generation Computer Systems*, vol. 108, pp. 385–397, 2020, doi: 10.1016/j.future.2020.02.033.

[20] S. Islam and A. Rahman, “Feature optimization in ML-based IDS for cloud environments,” *Journal of Network and Computer Applications*, vol. 173, p. 102896, 2020, doi: 10.1016/j.jnca.2020.102896.

[21] X. Zhang, Y. Chen, and H. Lin, “Deep self-taught learning for intrusion detection in cloud computing,” *IEEE Transactions on Cloud Computing*, vol. 10, no. 2, pp. 986–995, 2022, doi: 10.1109/TCC.2020.2998123.

[22] A. S. Ibrahim and M. S. Mahmud, “Random forest-based intrusion detection for cloud computing,” *IEEE Access*, vol. 8, pp. 135731–135742, 2020, doi: 10.1109/ACCESS.2020.3010342.

[23] R. R. Patel, D. Singh, and M. Kaur, “HTTP attack detection using ensemble-based machine learning models,” *IEEE Access*, vol. 10, pp. 9821–9833, 2022, doi: 10.1109/ACCESS.2022.3141546.

[24] C. Wang, F. Li, and S. Xu, “Hierarchical and distributed machine learning for scalable intrusion detection,” *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 362–375, 2022, doi: 10.1109/TIFS.2021.3129857.

[25] S. Kim and H. Kim, “Semi-supervised intrusion detection using generative adversarial networks,” *IEEE Access*, vol. 8, pp. 199524–199537, 2020, doi: 10.1109/ACCESS.2020.3035182.

[26] M. Ahmed, M. N. Islam, and S. Khan, “Hybrid semi-supervised learning for anomaly detection in cloud networks,” *Computers & Security*, vol. 113, p. 102567, 2022, doi: 10.1016/j.cose.2021.102567.

[27] P. Kumar, R. Tripathi, and S. Gupta, “Machine learning-based intrusion detection for software-defined networks,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1236–1247, 2021, doi: 10.1109/TNSM.2021.3053829.