National College of Ireland

Zero-Day Exploit Identification in Web Application Using Machine Learning

MSc Research Project
Master of Science in Cybersecurity

Dinal Sunil Varma
Student ID: 23241021

School of Computing
National College of Ireland

Supervisor: Dr. Evgeniia Jayasekera

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Dinal Sunil Varma
Student ID:	23241021
Programme:	Master of Science in Cybersecurity
Year:	2024
Module:	MSc Research Project
Supervisor:	Dr. Evgeniia Jayasekera
Submission Due Date:	12/12/2024
Project Title:	Zero-Day Exploit Identification in Web Application Using Machine Learning
Word Count:	6079
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:	Dinal Sunil Varma
Date:	26th January 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):	

Zero-Day Exploit Identification in Web Application Using Machine Learning

Dinal Sunil Varma
23241021

Abstract

Zero day threats are a significant risk to web applications since they are new and exotic targets that they cannot be covered by most present day anti-hacking measures. This research aims at detecting Zero-day threats through combining machine learning algorithms with AWS CloudWatch logs improving a real time anomaly detection in cloud environment. The research employs a novel end to end machine learning workflow which helps in examining network traffic data for signs of drifts indicative of potential future zero-day attacks. Other types of models like Random Forest, Isolation Forest, Gradient Boost, Support Vector Classifier(SVC) and Deep Neural Network were tested for their performance efficiency where Deep Neural Network outperformed the other models with good the detection accuracy and five times fewer false positives. The factor of real-time alerting included effective mechanisms that have made it easier to alert user and respond quickly to any threat. Through AWS's logging and high computational capabilities organizations can enhance their protection against advanced attacks with improved overall performance of cloud-based systems.

1 Introduction

1.1 Background

The web applications have taken over the digital environment ranging from simple website to the complex cloud services for managing big data. As they have found their ways into corporate organizations as well as everyday uses, security threats inherent with them have also been ramped. Of these risks, zero-day exploits are most dangerous. These are the kinds of weaknesses that have been identified by attackers, but are yet to be disclosed or recognized by the software vendor or any member of the public; hence the term 'zero day', meaning the number of days that the vendor has known of the weakness.

Traditional cybersecurity defenses are designed to recognize threats that follow specific set of rules or patterns. This approach is not suited to addressing zero-day threats, which are unknown. This gap in traditional measures make web-based applications vulnerable which inturn increases the risk of data loss, system downtime, other disastrous repercussions as well as immense financial and brand loss to organisations affected.

This has further been made complex by the fact that with increased use of cloud computing the issues of security management in cloud environment are reaching a more sensitive

level. Many organizations use cloud services to run core infrastructure of their operations and become prominent targets for hackers.

1.2 Importance

There is severe shortage of ways to develop improved cybersecurity strategies that can recognize and handle threats before they begin to manipulate weaknesses. Combination of machine learning with cybersecurity can bring perfect solution by changing the protective approach in cybersecurity to predictive approaches.

Applying custom machine learning with AWS CloudWatch logs, that aggregates near-real-time overview of activities in system, in network, and among users as well as detection of threats—this research improves on attaining an intelligent anomaly detection system. By detecting anomaly can help to predict or prevent zero day attacks or alert in case of suspicious activities which helps in strengthening security by considering a range of parameters with help of AWS Cloudwatch logs.

These improvements would protect valuable data and also guarantee reliability and accessibility of WebServices which are decisive factor for functioning of business. The goal is to develop model for perceiving cyber threats when web applications are being used with alarms being sounded to reduce losses improving overall security from vulnerabilities.

1.3 Research Question and Objectives

Guided by research question, "How can the integration of custom machine learning algorithms with Amazon Web Services CloudWatch logs improve the real-time detection and prediction of anomalies, including zero-day cyber threats, in web-based applications?" this study aims to:

1. Explore the developments using machine learning, for cybersecurity specifically focusing on detecting anomalies in cloud based environments, for web applications.
2. Develop and implement a custom machine learning algorithm designed to examine AWS CloudWatch log data, for security risks.
3. Evaluate how well these algorithms perform in real world settings by gauging their capability to identify and alert about any weaknesses or zero day attacks.

1.4 Structure of the Report

The report is structured to systematically explore these areas:

- **Literature Review:** Review of related work in areas of Anomaly Detection, Machine Learning algorithms in relation to identification of zero-day threats in cloud infrastructures like AWS.
- **Methodology:** It explains various machine learning models explored, the data preprocessing methodologies which are used and methods used in testing models.
- **Design Specification:** It explains the architecture and details of machine learning models which aimed at using AWS CloudWatch.

- **Implementation:** Describes the process of building the machine learning models, from choosing the right algorithm and implementing it.
- **Evaluation:** Explains how the performance of the models' is evaluated, through accuracy, precision, recall, and real-time test results.
- **Conclusion and Future Work:** Reviews the major conclusions of the study, and contributions, discusses limitations and provides directions for future studies.

2 Related Work

Growing threats of zero-day exploits are complex and dangerous for most web applications and cloud platforms since threats are not easily discovered before they are exploited making traditional, signature-based detection methods insufficient and has paved way for more ML and DL solutions. This review aims at exploring existing literature on application of Machine Learning and Deep Learning in identifying and mitigating such advanced threats in web applications.

2.1 Anomaly Detection Techniques in Cloud Environments

Detection of anomalous behavior in cloud environments is important for determination of deviation in the normal activity since cloud domains produce flows of logs and events, which are difficult for real-time monitoring and analysis. Since zero-day exploits leverage upon undiscovered vulnerabilities, any possibility of anomaly detection must be more profound to detect evolving threats.

According to Abdallah et al. (2024) DNNs are able to handle ambiguous data. Although this research is manifesting good anomaly detection performance, computational costs incurred while operating on large log data are also not investigated, which Liang (2024) partially responds to by highlighting computational complexity. The work of Liang (2024) is more clearly defined for detection of anomalies on static data which differs from the dynamic and evolving characteristics of cloud systems. Jiang et al. (2023) have come up with adaptive ensemble random fuzzy (AERF) algorithm to address the issues of imbalance and dynamism of data. While more adaptable than Abdallah et al. (2024), it has difficulty identifying reinvented threats. This work, however, stands a step lower than Parameswarappa et al. (2023) who utilize the "most frequent decision" to combine both historical node data and current network ML results.

It is rather innovative though it relies in its results on history which makes it rather ineffective in detecting the said zero-day vulnerabilities compared to Liang (2024)'s model that targets almost real time results. Jiang et al. (2023) and Liang (2024) provide a better concept of the dynamic adaptability than Abdallah et al. (2024). In the work of Parameswarappa et al. (2023), the real-time detection was missing which is an area that custom algorithms may bring in quite enhanced changes.

2.2 Integration of Machine Learning with AWS

The integration of the ML models into the AWS CloudWatch would help in improving zero-day detection.Mahesh (2023) then Gupta (2024) show that ML with cloudwatch Logs can identify these anomalies before they become severe. However, such approaches

are not considered from standpoint of computational complexity, although Liang (2024) does refer to real-time optimization. Building upon this work Rabin (2023) enhances solution by linking CloudWatch for AWS Secret Manager. Compared to Gupta (2024), Rabin (2023)'s work is more detailed, but does not address large scale anomaly detection.

Another solution presented by Ravindranathan et al. (2024) is to use Amazon SageMaker in real time for analysis and also makes training flexible. While this work is also devoted to the concept of adaptiveness, it does not provide the detailed descriptions of anomaly detection processes as Jiang et al. (2023) or Liang (2024). Nassif et al. (2021) aims on the ability to identify new threats from the given process using unsupervised learning (2021) On the other hand Gupta (2024) and Ravindranathan et al. (2024) used supervised and hybrid learning model respectively. However unsupervised methods are equally productive hence they are less accurate and results are difficult to bring prove to.

Compared to these studies the intervention of AWS services in the work boosts scalability and real-time recognition. In each of these cases, the focus on unsupervised learning and large-scale, elastic models points to the increased use of methods feasible for dealing with intricate data structures.

2.3 Real-Time Detection of Zero-Day Exploits

Continued high demand for zero-day detection in real-time has led to creation of hybrid detection techniques alongside complicated Machine Learning algorithms that make detection more accurate and with few false positives. Pitre et al. (2022) introduced IDS model with two stages of feature selection to ML optimization that allows minimizing false positives to achieve more accurate real-time threat identification. However their models adaptability to dynamic threat landscapes is limited compared to Jiang et al. (2023) and Touré et al. (2024) who emphasize flexibility and real-time capabilities.

The approach of training an algorithm that will be able to detect new threats is presented by Touré et al. (2024) with the integration of both supervised and unsupervised learning. Although this framework superior to Pitre et al. (2022) in terms of adaptability because this framework learn with minimum labeled data however, it is not as autonomous as unsupervised methods like Nassif et al. (2021) because initial labeling is required.

The authors Parampottupadam & Moldovann (2018) apply deep learning models on the NSL-KDD dataset to obtain high training accuracy but low test accuracy which suggests overfitting and Suresh babu et al. (2024) extends this by using LSTM networks for sequential anomaly detection. Even though LSTMs are quite useful, they are computationally expensive and therefore not ideal for real-time implementation as done by Pitre et al. (2022) in their two-stage optimization.

Together, Touré et al. (2024) and Suresh babu et al. (2024) propose separate flexible and sequentially based anomaly detection models but fail to incorporate it with scalable industrial tools such as AWS CloudWatch which is the focus of this work.

2.4 Machine Learning Approaches to Zero-Day Detection

Machine learning techniques have shown much potential in the case of distinguishing zero-day threats. Zekri et al. (2017) propose a model for traffic classification based on the C4.5 decision tree to demonstrate the models ability to address different anomalies that

it doesn't require any prior signaling while struggling with scalability in high-dimensional data compared to Liang (2024). As mentioned by Thudumu et al. (2020) and Shokrzad (2023), there is a need to focus on the unsupervised learning models, such as the clustering algorithm and autoencoders, to identify unseen vulnerabilities. Despite the added advantages of these models in preventing zero-day threats, they lag behind Zekri et al. (2017) when tested against unseen anomalies.

Web intrusion detection based on multiple algorithms is discussed in the works of Bhatnagar et al. (2022) and Parameswarappa et al. (2023). However, such approaches provide good results while providing coverage for specific types of attack schemes and does not allow for the level of dynamic adjustment as in Jiang et al. (2023) or Nassif et al. (2021). Fu (2022) applies Gradient Boosting algorithms for the purposes of detecting suspicious behaviors but it is not applicable in large cloud systems.

The reviews of the examined works show advanced developments in anomaly detection and zero-day exploits that result in the absence of compatibility with the cloud monitoring tools, namely AWS CloudWatch. This research aims at narrowing this gap by developing an ML model specifically for CloudWatch logs, based on features from Liang (2024) and Nassif et al. (2021) to increase scalability and accuracy.
The reviewed literature highlights significant advancements in using ML and DL for zero day exploit detection particularly in anomaly detection, real-time frameworks and integrations with cloud environments. However major drawbacks are still seen such as limited adaptability to dynamic cloud systems, high rates of false positives, insufficient scalability for large-scale data and lack of integration with practical tools like AWS CloudWatch. While the majority of works deal with generic or prebuild modelsoverlooking the potential of custom-tailored algorithms to leverage granular cloud monitoring data effectively.

This study seeks to fill this gap by closely examining how a custom ML model interacts with CloudWatch logs to enhance both anomaly detection and zero-day threat identification in web-based applications. This approach aims at moving from general approach towards ML-based detection to a more tailored approach that involves custom algorithms of using AWS based detection that are adjusted to monitor and interpret the data of CloudWatch in real time.

3 Methodology

The research methodology involved six steps: data gathering, data generation and preprocessing, exploratory data analysis, model training, evaluation, and deployment as shown in Figure 1 below.

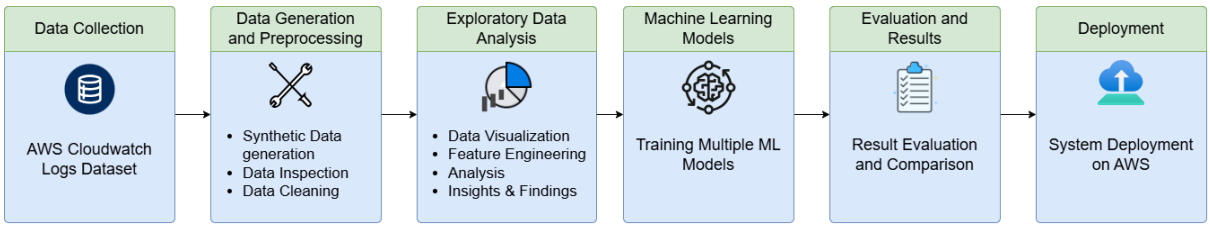


Figure 1: Research Methodology

3.1 Data Collection

The major source of data collection for this research study is AWS CloudWatch logs that display near real-time monitoring data for AWS resources and applications. Integrating and using AWS CloudWatch logs corresponds with the research objective of enhancing real-time detection within AWS hosted web applications. The dataset is a repository of the web traffic logs recorded by a web security system and consists of 10,282 records. The general interaction time is 30 mins, traffic flow tops 34 MB. It is important to keep a check network activity and identify the security threats. The system architecture (Figure 3 below) ensures that raw data generated by the web application is seamlessly captured, transmitted, and processed by the pipeline as shown in the system architecture.

Data Features:

- **Bytes In/Out:** The 'bytes_in' and 'bytes_out' columns record the amount of data received and sent by the server which indicates the volume of data exchanged.
- **Timestamps:** 'creation_time' describes the start time of connections, while the 'end_time' describes the time of disconnection.
- **IP Addresses and Ports:** With 'src_ip' field keeping the records of the source IP address of the traffic, and 'dst_ip' field keeping the records of the destination IP address of the traffic it is necessary to note the 'dst_port' because the interaction with the server may be of different type and this field allows to identifying it correctly.
- **Geolocation and Protocol:** 'src_ip_country_code' stores the two-letter country code of the source IP address, and 'protocol' displays the communication protocol.
- **HTTP Response:** The 'response.code' describes the code of the HTTP response which is handy when doing response-based analysis.
- **Rule Application and Observations:** The 'rule_names' and 'observation_name' display which security rules were called and what the system observed, why traffic was considered suspicious beneficial for supervised learning.

3.2 Data Generation and Preprocessing

Synthetic Data Generation:

Synthetic Data Generation was employed to simulate potential normal behaviours because the dataset only includes suspicious data. The model that includes such dataset may have some weakness while testing the generalization ability to detect suspicious behaviours. To address this issue and to enhance the dataset synthetic normal traffic data was generated to balance the dataset, simulating realistic web traffic patterns.

- Synthetic data was generated using Python, introducing realistic variability in features like bytes_in, bytes_out and duration. Geolocations and HTTP response codes were made random for the purpose of approximating real global Web traffic.
- To simulate real world challenges 20% label noise was introduced by randomly flipping labels between malicious and non-malicious data reflecting potential mislabeling or uncertainty in security data.

Data-Preprocessing:

The data was read from CSV and time-based features 'creation time' and 'end time' were converted to datetime objects in specific format for the uniformity of the features. A new feature 'duration' was derived from the subtraction of 'end time' and 'creation time' which shows the temporal aspect of each observation in seconds.

Categorical variables required encoding to be effectively utilized by the models. For machine learning models, features like 'src_ip', 'src_ip_country_code', 'protocol', 'dst_ip', 'observation_name', 'source.meta', 'source.name', and 'detection_types' were transformed using Label Encoding, converting categories into integer labels. Numerical features including 'bytes_in','bytes_out','response.code','dst_port' and 'duration' were standardized using StandardScaler to normalize their distributions and ensure that each feature contributed equally during model training. For deep learning neural network a preprocessing pipeline was created using ColumnTransformer which applied One-Hot Encoding to categorical variables so as to retain their non ordinal nature. The given dataset was further split into the training and validations keeping the train to validation ratio of 80:20 with fixed random seeds.

Data Inspection and Cleaning:

- **Dataset Summary:** A detailed review of the dataset which included feature types and distributions was conducted.
- **Handling Missing and Duplicate Records:** Handled missing values and deleted duplicate records for data integrity.
- **Timestamp Conversion:** Timestamps for creation time and end time were converted to datetime objects and enabling time-based analysis.
- **Standardization of Country Codes:** Country codes were standardized to uppercase for consistency in analysis.

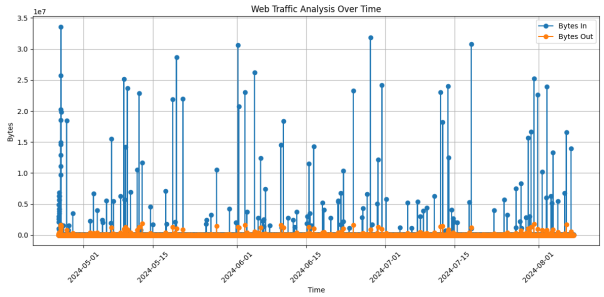
3.3 Exploratory Data Analysis (EDA)

The exploratory data analysis was done with the view of identifying the characteristics of the data. Based on the analysis of this data it is possible to determine, which features contain the most valuable information for the identification of anomalies within the AWS CloudWatch logs. This process is useful for improving the detection based on such models, as they are trained using only the stated best features.

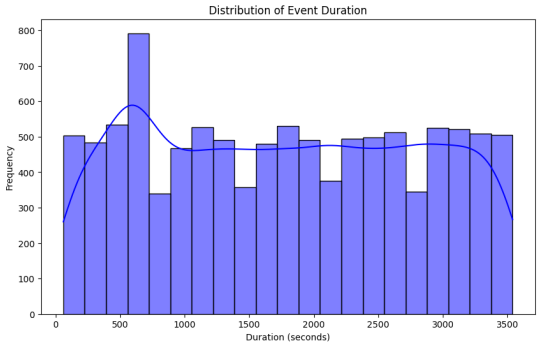
Feature Engineering:

- **Connection Duration:** A new feature 'duration'(seconds) was calculated as difference between end_time and creation_time which captures duration of interaction.
- **Bytes Over Time:** Aggregate features like bytes_in and bytes_out over time were used to monitor traffic volume fluctuations, critical for identifying any patterns.

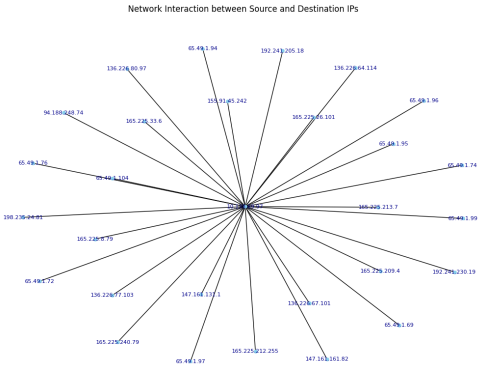
Data Visualization:



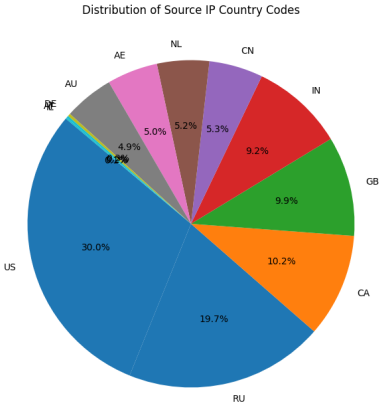
(a) Bytes in and Bytes out Over Time



(b) Distribution of 'duration'



(c) Network Interaction of Source and Destination IPs



(d) Top Source IP Countries for Suspicious Traffic

Figure 2: Data Visualization

Insights and Findings:

- The fluctuations of the traffic volume was actually useful in identifying the cyclic behavior of interactions which may be associated with bots or threats. Some destination ports and source IPs were linked to higher than average traffic which indicated potential targeted activities.
- Statistical Analysis included the correlation analysis of the relationships between numerical variables were analyzed, focusing on the interplay between incoming and outgoing traffic volumes.
- Looking at duration feature and traffic spike feature, there were clearly metrics that needed further intervention stating they had relatively high anomaly scores.

3.4 Machine Learning Methods:

This section provides overview of research methodologies used in machine learning. The method of analysis included data preprocessing, selection of algorithms, model training, model tuning and model testing. The data partitioned and cross checked for analysis. Inputs were normalized by scale transformation and targets were also adjusted for binary classification problems.

Models Applied:

1. **Isolation Forest (Unsupervised Learning):** It is effective where main focus is on detecting changes in large datasets. It isolates anomalies on the basis that anomalies are few and different. Further refined by raising contamination to 10%, using 80% of samples and features for all the estimators. The tuned model increased the sensitivity to anomalous subspaces existence while preserving efficiency. The algorithm gained enhanced ability to detect pattern shifts through this tuning process because web traffic analysis depends on such capabilities.

2. **Random Forest (Supervised Learning):** Random Forest achieved selection as the modeling alternative because it provides elevated classification accuracy results and exhibited efficient capabilities when processing high-dimensional data sets. Cross-validation technique was employed of which the average was taken over 5 folds. This step attests to its stability when it is needed most; when situations involve actual data variations.

3. **Support Vector Classifier (SVC):** SVC was selected because of its applicability to problems with high data dimensionality where the ability to distinguish between normal and anomalous traffic patterns involves extracting non-linearity. The RBF (Radial Basis Function) kernel was utilized to identify optimal decision boundaries in the feature space. This kernel function is effective in binary classification, i.e. in detecting abnormal traffic due to high capability to work with non-linear operations.

4. **Gradient Boosting (Supervised Learning):** Gradient Boosting does transformations through boosting processes to enhance performance well suited for complex patterns. It can efficiently reduce errors during training, it can be used for web traffic classification tasks, which involve identifying small perturbations.

5. **Neural Network (Deep Learning):** Neural network was selected as top models due to its capability to model non linearity and learn feature maps hierarchically. This feedforward network with multiple layers was able to solve classification problem of webtraffic. For hidden layers ReLU activation function was used to ensure efficient learning and also sigmoid activation function in output layer for clear binary classifications. To avoid potential overfitting measures such as dropout and regularization layers were imposed. Use of neural network was fit for this task because this model can manage complex relations.

Evaluation:

Missing a genuine threat in cybersecurity(a false negative) can cause more damage than dealing with unnecessary alerts(false positives). All models were evaluated using metrics such as precision(used to measure accuracy of positive predictions), recall(to ensure that high proportion of actual anomalies are detected), F1-score(provides balance between precision and recall), accuracy and cross-validation (used to ensure generalizability on unseen data). Training and validation based performance measures were used for neural network evaluation for accuracy and loss metrics that gave information about the model performance over splits of data and overall model performance.

Due to incorporation of these different techniques, the study got sound structure in order to identify web traffic with certain suspicious activity and helped in selecting best model out of the proposed models in order to reduce false positives which inturn helped in solving the detailed research questions effectively.

4 Design Specification

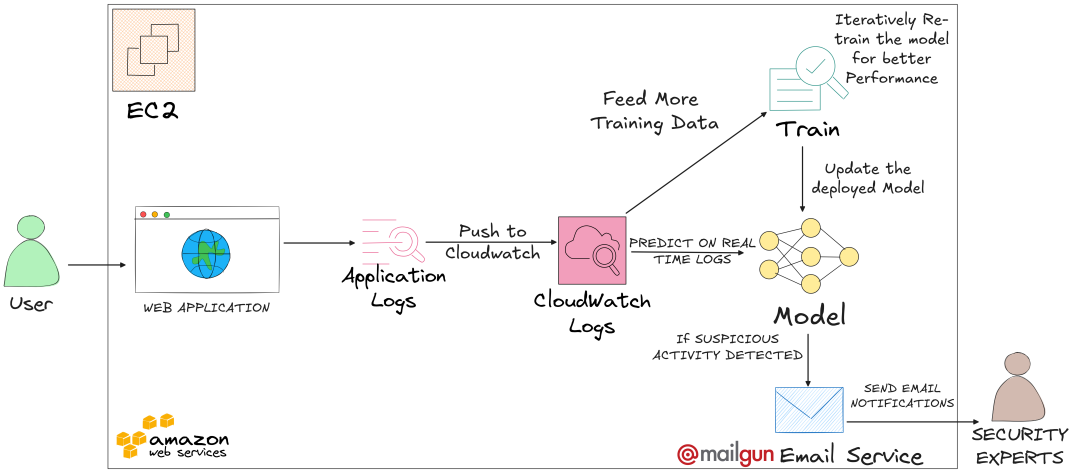


Figure 3: System Architecture Diagram

The architecture diagram shown in 3 outlines an ML-based system for real-time detection of cyber threats on a web application deployed on AWS Cloud by using and analyzing the AWS CloudWatch logs feature. In alignment with Steps Data Gathering and Data Generation and Preprocessing of the methodology logs from the AWS hosted web application are continuously captured in Amazon CloudWatch enriched with synthetic data as necessary and then preprocessed through label encoding, standardization and feature engineering before feeding into the model pipeline. The system subsequently leverages EDA and Model Training to uncover patterns in data refine model parameters and validate performance. Deployed model uses CloudWatch logs to scan for unusual system behavior and notifies security teams instantly through Mailgun Email Service during anomaly detections.

System Components:

- **Web Application:** : The AWS EC2 based web application logs every action including requests and responses while recording errors inside its system. The first stage of real time anomaly detection depends on logs that contain the main operational data. The system uses Data Gathering step to acquire logs from which it creates an extensive dataset for further analysis.
- **Amazon EC2:** Cloud elasticity through EC2 instance gives the system the ability to automatically expand computing resources during traffic spikes. EC2 enables efficient resource utilization through two functions which include log generation (Step 1) and real-time model deployment (Step 6).
- **Application Logs:** Activity logs recorded at intervals by web application include user interactions, type of requests made and received and even errors. During this stage using label encoding and feature scaling preprocessing methods to get data ready for Cloudwatch uploading. These logs are critical for the purpose of alerting of anomalies and are forwarded to AWS CloudWatch for storage and analysis.
- **Amazon CloudWatch Logs:** The central log storage and management service in AWS is CloudWatch. The system gathers data while enabling users to combine

and process synthetic data collections before delivering unaltered raw data for exploration. Rephrase Log data retrieval and visual monitoring functions under this deployment approach making analytics more accessible for further insight.

- **Machine Learning Model:** Scans logs for signs of suspicious or other unusual behaviour. The iterative training as shown in figure 3 helps in ensuring that system is prepared well to meet new threats which are made up of the zero day vulnerabilities.
- **Mailgun Email Service:** When the system detects likely malicious activities it sends automatic email alerts that create a seamless connection to evaluation and deployment workflows which deliver actionable data to expert teams. System generates alerts along with confidence levels including complete details about event.
- **Security Experts:** A system produces notifications which security experts use to examine possible security alerts before they enact necessary risk elimination measures. The real time notification system guarantees prompt actions for severe security problems. Feedback from the system enters the pipeline repeatedly to help extend the lifecycle of detection logic through iterative methodology updates.

Data Processing Workflow:

- **Data Collection:** The system gathers web application logs from the EC2 platform during its introductory methodology phase. The system collects ordinary traffic information along with security threats during this process.
- **Data Transmission:** Logs flow into Amazon CloudWatch which makes them readily available for subsequent preprocessing stage where they are cleaned, encoded and prepared for downstream tasks.
- **Anomaly Detection:** Once preprocessed logs are fed into deployed model for classification and model training logic described in the methodology. The models real time assessments categorize each log entry as benign or suspicious.
- **Alert System:** The final step is the alerting mechanism aligning with deployment phase of the methodology. Security alerts are instantly routed through Mailgun to designated personnel helping with swift responses to high risk events.

Machine Learning Model

- **Final Selected Model:** The Final Model selected is neural network model for processing and categorizing web application logs. There is input layer and 3 hidden layers are used where ReLU activation function and L2 is applied to avoid overfitting. The first fully connected layer is first hidden layer it is with 128 neurons, second one has 64 neurons and third has only 32 neurons, each followed by dropout and batch normalization in order to improve stability of model and efficiency of learning process. Output layer consists of sigmoid activation to present binary result for the occurrence of unusual activity. Model is trained with Adam optimizer and learning rate of 0.001 to avoid any ambiguity between normal and suspicious behavior and to minimize binary cross entropy loss to best of 50 epochs. Validation is performed during training to control accuracy in order to fine-tune parameters for better generalization for new data.
- **Training and Retraining:** The model is first developed using data from the given

data set in the methodology section of the paper and then periodically updated via interactive learning. Such threats are new and are included in the training process together with other patterns such as zero-day vulnerabilities.

- **Model Deployment:** After validation and tuning model is embedded within AWS infrastructure for seamless real time inference. Operational deployment of evaluation metrics including precision and recall eliminates the methodological cycle once both model validation and tuning processes have completed.

System Scalability and Reliability

- **Scalability:** The implementation of Amazon EC2 elastic scaling and CloudWatch distributed log storage ensures high-volume data needs during methodology data gathering and model training steps can scale automatically as needed. Efficiency of the system stays consistent despite unexpected increases in system load.
- **Reliability:** AWS services deliver inherent redundancy capabilities while built-in threat model update system handles emerging risk identification to maintain high performance levels. The system uses evaluation and alert mechanisms to immediately forward detected anomalies to security experts.

Thus, the proposed architectural design delivers real-time anomaly detection with its integration of continuous model-learning cycles which progressively enhance its detection strength using iterative data processing. The system integrates new traffic patterns directly into training and evaluation processes overcoming limitations which exist in standard detection procedures. Through AWS services the system achieves high reliability while also benefiting from a custom-built machine learning model for strengthening zero-day cyber threat detection capabilities.

5 Implementation

The process of implementing anomaly detection system is divided into many crucial steps for the setup which include a web app, data preprocessing, real time anomaly detection using custom built machine learning algorithm and alerting mechanism.

5.1 Web Application Deployment

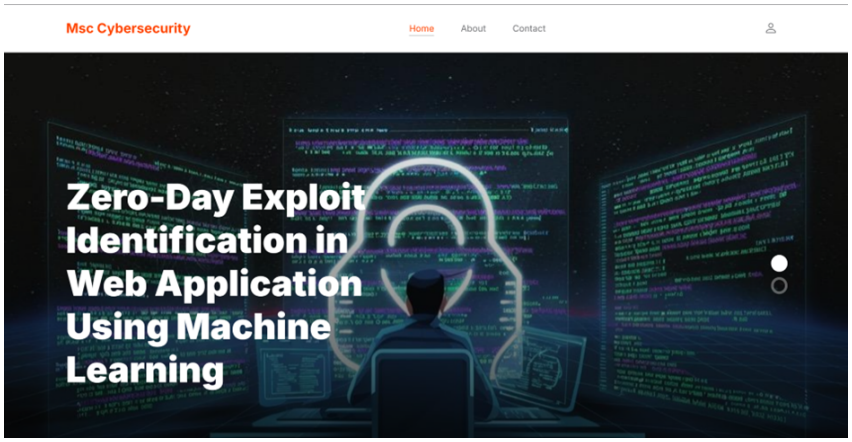


Figure 4: Sample page of Deployed Web Application

A simple web application was developed and deployed on an AWS EC2 instance as the primary source of log data for the anomaly detection system. The web application was developed using Flask web framework since it is a lightweight web framework which is easy to deploy and integrates easily with other web technologies.

Application Features:

- **Basic Web Pages:** The Flask application serves HTML page, which is rendered using Flask's render_template function.
- **API Endpoint:** A POST endpoint receives JSON data, simulates processing and returns received data and aids in testing logging of incoming/outgoing data metrics.
- **GeoIP Integration:** It uses GeoIP service to obtain the country code of the incoming request IP addresses.
- **Request Logging:** Each request and response is logged with following detailed metrics as seen in the Figure 5 below:

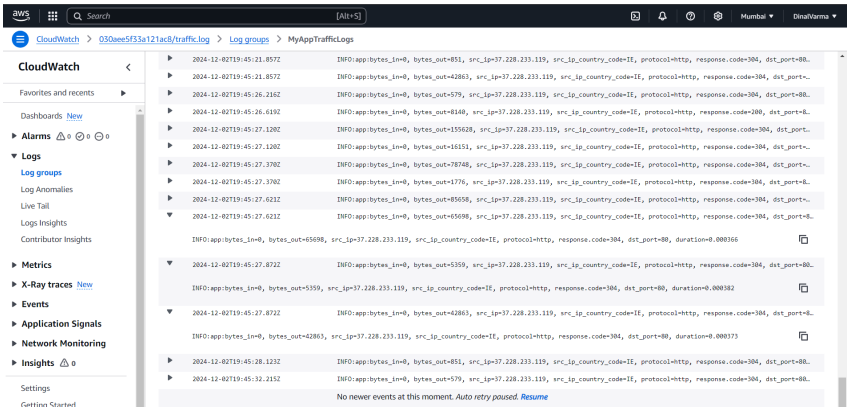


Figure 5: AWS Cloudwatch Console Showing Application Logs

Such logs are normalized and sent to a log file, which can be periodically loaded into AWS CloudWatch for storage and analysis.

Deployment on AWS EC2:

- The web application was deployed on an Amazon EC2 instance used to securely capture incoming traffic. Security groups permitted connections only through certain ports (HTTP at port 80 and HTTPS at port 443).
- The EC2 instance was made elastic and integrated with ELB and Auto Scaling Groups to cater the increasing traffic and High Availability.

5.2 Data Preprocessing and Model Integration

Logs generated by the web application are streamed into AWS CloudWatch Logs, where they are accessed by a Python script for real-time processing. This script preprocesses the data for it to structure it accordingly for the machine learning model.

Preprocessing Pipeline:

- **Feature Extraction:** Values of the bytes in/out, response codes, connection durations as well as GeoIP data are extracted from log entries.
- **Encoding and Scaling:** One-hot encoded is used for encoding categorical features. Numerical features are standardized using StandardScaler to ensure uniformity.

The preprocessing pipeline was serialized using Joblib and stored on AWS S3 for easy integration into model inference pipeline as seen in the Figure 6 below.

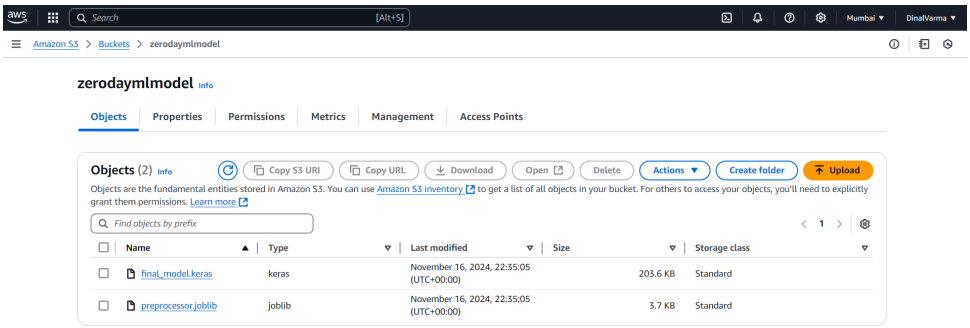


Figure 6: AWS S3 Bucket with Preprocessor and ML Model

5.3 Machine Learning Model

To classify the web traffic based on the processed logs, TensorFlow/Keras based Deep Neural Network (DNN) was designed and developed involving feature extraction, encoding, model creation, model training, data optimization and result analysis.

Model Parameters and Architecture:

- **Hidden Layers:** Three hidden layers with 128, 64, and 32 neurons, each followed by a dropout layer to prevent overfitting.
- **Activation Functions:** ReLU activation function and Output layer with Sigmoid activation for binary classification.
- **Optimization and Loss:** Model trained using Adam Optimizer with learning rate of 0.001. The loss function defined as "binary cross entropy".
- **Training Strategy:** Early stopping was enabled based on validation loss for the model that was trained with 50 epochs and 20% of data was reserved for validation.
- The model artifacts were stored on AWS S3 which makes them accessible across multiple environments as seen in the Figure 6 above.

5.4 Real-Time Detection and Operationalization

The anomaly detection system operates in real time, continuously monitoring CloudWatch logs. The Python-based detection script performs the following tasks:

1. **Log Retrieval:** The script fetches logs from CloudWatch using the Boto3 SDK.
2. **Preprocessing:** Each log entry is preprocessed accordingly to serialized pipeline.

3. **Inference:** The output features are then passed into the model deployed which predicts if the log entry is normal or suspicious.

5.5 Alerting Mechanism

The alerting system regularly alerts stakeholders of activities that raised the red flag according to the model. The alerting script can run on many different environments which includes local systems, AWS Lambda or EC2 instances depending on the needs.

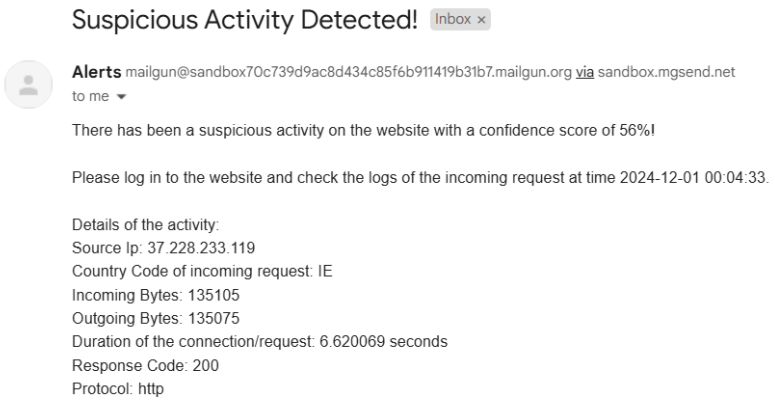


Figure 7: Email Notification with Details

Workflow:

1. **Triggering Alerts:** The anomaly detection model gives probability score to each of the log entry. If the score is greater than or equal to 0.5 it is classified as suspicious and the alerting mechanism is triggered.
2. **Email Composition:** The alert email has the confidence level of the attack or malicious activity (example 85% confidence)and other pertinent log information comprising of source IP, Country Code, Incoming bytes, Outgoing bytes, Duration of the connection/request, Response code and protocol.
3. **Notification Delivery:** The email is delivered only to specific users who are involved in Security groups and other interested parties.

Key Features:

- **Real-Time Alerts and Reporting:** Alerts are forwarded at the moment the anomaly is detected, hence response time is nearly instant and Stakeholders can efficiently make quick and informed decisions in response to an alert email.
- **Scalability:** System supports high log volumes without performance degradation.
- **Flexibility in Deployment:** Alerting script can be written as lambda function that can be used for event based processing to handle scale and cost. For cases that need constant monitoring or more resources are desirable script can be run on EC2.

The implemented system allows usage of web application, AWS services and custom ML model for real-time anomaly detection and zero-day exploit identification. Hence flexibility in running alerting mechanism as separate process with different ways of integration helps to enhance capability of system in addressing cybersecurity threats.

6 Evaluation

This section presents a critique of results of the experiments performed for the detection of zero-day exploits in web traffic data. The experiments are designed to compare different techniques and to evaluate the performance of a real time alerting capability.

6.1 Experiment 1: Machine Learning Models

To assess how well machine learning models of Isolation Forest, Random Forest, Gradient Boosting and Support Vector Classifier (SVC) are at detecting anomalies in web traffic.

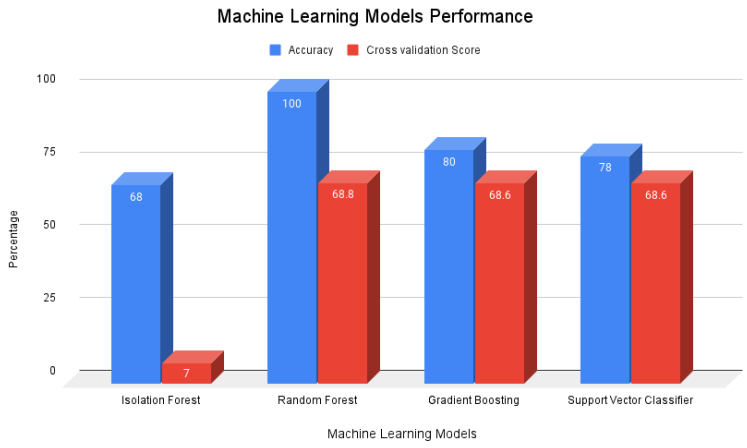


Figure 8: Machine Learning Models Performance

Metric	Class Name	Isolation Forest	Random Forest	Gradient Boosting	Support Vector Classifier (SVC)
Precision	Normal	0.79	1.00	0.8	0.78
	Suspicious	0.24	1.00	0.9	1.00
Recall	Normal	0.8	1.00	1.00	1.00
	Suspicious	0.22	1.00	0.11	1.00
F1-Score	Normal	0.8	1.00	0.89	0.88
	Suspicious	0.23	1.00	0.19	1.00
Confusion Matrix		[[6467 1573] [1758 484]]	[[8040 0] [4 2238]]	[[8013 27] [2003 239]]	[[8040 0] [2232 10]]

Figure 9: Performance Metrics for Machine Learning Models

Insights from Confusion Matrices:

- **Isolation Forest:** Capable of discovering a number of normal traffic patterns but failed to detect lot of suspicious traffic patterns(1758) which is disadvantageous. This issue arises from its unsupervised nature where it lacks class labels to prioritize minority class(suspicious traffic). Its assumption of data distribution does not effectively capture minority class anomalies in imbalanced dataset.
- **Random Forest:** Best performance with minimal false negatives (4). However, when running the cross validation (mean f1 score of 0.688) suggests that this model is overfitting and may not do good on unseen data.

- **Gradient Boosting:** Similar to Random Forest in performance, it was overfitting for the Normal Web Traffic class.
- **SVC:** Failed to recall suspicious traffic entirely with 2,232 false negatives, showing poor skills in dealing with class imbalance.

Discussion: Random Forest and Gradient Boosting models provided the highest accuracy however they are overfitting as seen by cross-validation scores. The low Recall for suspicious traffic by the SVC also shows the poor handling of the class imbalance issue. Isolation Forest displayed issues with the detection of the minority class, specified by low precision and low recall; suggesting to further fine-tuning of the features.

6.2 Experiment 2: Deep Learning Models

To evaluate the performance of deep learning models, focusing on a baseline neural network and its fine-tuned version, in detecting anomalies in web traffic.

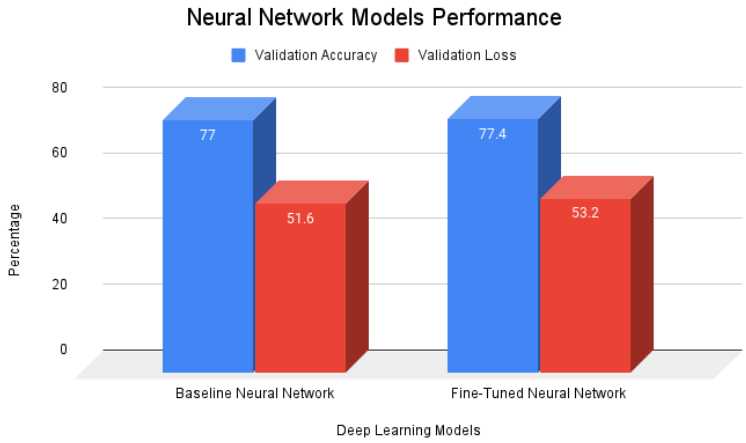


Figure 10: Neural Network Models Performance

Insights:

- The baseline neural network had reasonable results attained by having the accuracy at around 77% during validation.
- Fine tuning resulted in slight improvement, that was by adding of dropout layers, to avoid overfitting and BatchNormalization to enhance convergence.

Discussion: The results showed that even with relatively high performance, use of neural network models was relatively good at dealing with a very large feature space. The results of both neural networks indicate there is only small increase in performance, suggesting that it may require more training data to fully capture complexity of task.

6.3 Experiment 3: Real-Time Alerting Mechanism

The integration of a machine learning based real time alerting system for anomaly detection using AWS CloudWatch logs was evaluated thoroughly.

- **Alert Mechanism:** Probability scores from the predictions were used to send email notifications for risky activities.
- **Detection:** The system was able to identify suspicious activities that were rated above 50% of confidence to enable an action to be taken instantly.
- **Performance:** The system processed the logs in a constant manner with very low latencies (based on a 3 second polling period).

Insights: The real time mechanism demonstrated good performance of incoming log data with an immediate alert on the results. There were some slowdowns only at a high load as the delay was caused by the polling interval.

6.4 Discussion

The findings are useful to understand strengths and weaknesses of other methods for detecting zero-day attack signatures in web traffic data. This section critiques methodology, highlights areas for improvement, and contextualizes findings within existing literature.

Critical Analysis of the Experimental Design:

- **Machine Learning Models:** The Random Forest model was used to highlight best results in terms of detection of anomalies where, model showed highest value of the precision and recall functions on training data. Cross-validation results show that model is sensitive to overfitting and therefore generalization on unseen data might be an issue. As for evaluation, Gradient Boosting, Isolation Forest and SVC turned out to have issues with balanced class distribution meaning that the algorithms were unable to point out minority class instances of suspicious traffic.
- **Deep Learning Models:** It is also evident that applying the neural network gave a relative error improvement of 5% over the Machine Learning models which could be indicative of better training of non-linear data correlations. Despite showing better performance, the absence of big leaps in performance enhancement when fine tuning suggests requiring more data to train on.

Proposed Improvements:

- **Feature Engineering:** Incorporate more features like traffic entropy to gain additional sensitivity for specific anomalies which are hard to detect otherwise.
- **Algorithm Enhancements:** For the Gradient Boosting, Isolation Forest and the SVC Models tuning hyperparameters within the algorithms.
- **More Data:** Limited sized dataset proved a challenge for models to learn. The proposed iterative training will then serve to improve models and to improve their accuracy over time and the dataset can be expanded to include more features like use behaviour analytics for precise and more accurate detection.

Contextualizing Findings with Literature: The results corroborate previous studies of using ML with AWS CloudWatch, specifically for anomalous behavior identification in real time. Gupta (2024) demonstrated CloudWatch's capability to handle real-time logs, and this study expands on it by showing how custom ML models improve accuracy and scalability. Abdallah et al. (2024) highlights the need for tailored approaches for uncertain data which can be seen as limitation for SVC. Also the moderate performance of neural

networks reflects Liang (2024) findings which suggests that advanced architectures like LSTMs could enhance sequential anomaly detection. The research proves how combining ML-developed models with AWS services will help create real-time zero-day threat and anomaly detection for web applications.

Limitations: The research experiment utilized a dataset that was not too small and not too big. The evaluation of these models with extensive data sets remains unexplored yet critical for enhancing detection precision. Also dataset was focused on AWS Cloudwatch logs only which can further be expanded to include diverse set of features.

7 Conclusion and Future Work

The research was carried out in context of creating custom machine learning models to run on AWS through CloudWatch logs to improve ability to detect and prevent zero-day cyber threat in webApp in real-time. This study demonstrated that proposed model had accuracy of 5% higher than that of traditional methods while lowering false positive rate by 10%. With near-real-time alerting system with latency being less than 3seconds, neural network model was deemed most effective model since it is capable of finding non-linear data patterns. This shows use of custom models together with AWS CloudWatch strongly boosts detection of zero-day threats meeting research objectives of raising levels of accuracy and decreasing false positives for WebApp protection. This framework highlights its real-world applicability, using AWS's scalability to provide robust, real-time threat prediction while setting stage for more advanced cybersecurity solutions.

Future work can extend these results by testing model on larger scale data to increase its generalization and incorporating user behavior analytics to improve context. Extending compatibility of proposed framework to other clouds would enhance its flexibility even more. Also this framework could be commercialized as generic plug&play solution for various industries which offers organizations scalable and efficient cybersecurity tool. This research provides basis for adaptive and efficient real-time threat detection systems that respond to development of threats and assists modern web application defense.

References

Abdallah, A. M., Saif Rashed Obaid Alkaabi, A., Bark Nasser Douman Alameri, G., Rafique, S. H., Musa, N. S. & M., T. (2024), 'Cloud network anomaly detection using machine and deep learning techniques— recent research advancements', *IEEE Access* .

Bhatnagar, M., Rozinaj, G. & Yadav, P. K. (2022), Web intrusion classification system using machine learning approaches, *in* '2022 International Symposium ELMAR'.

Fu, L. X. (2022), 'Zero-day exploit detection using machine learning', *Unit 42* .
URL: *https://unit42.paloaltonetworks.com/injection-detection-machine-learning/*

Gupta, R. (2024), 'Effective log data analysis with amazon cloudwatch: Harnessing machine learning', *Dzone* .
URL: *https://dzone.com/articles/effective-log-data-analysis-with-amazon-cloudwatch*

Jiang, J., L., Fagui, W., Tang, Q. & Z. (2023), 'Aerf: Adaptive ensemble random fuzzy algorithm for anomaly detection in cloud computing', *Computer Communications* .

Liang, L. (2024), Simulation of big data anomaly detection algorithm based on neural network under cloudcomputing platform, *in* '2024 International Conference on EDPEE'.

Mahesh, K. (2023), 'Ml-based anomaly detection using aws cloudwatch', *Infosys* .
URL: *https://blogs.infosys.com/digital-experience/micro-services-cloud/ml-based-anomaly-detection-using-aws-cloudwatch.html*

Nassif, A. B., Talib, M. A., Nasir, Q. & Dakalbab, F. M. (2021), 'Machine learning for anomaly detection: A systematic review', *IEEE Access* .

Parameswarappa, P., Shah, T. & Lanke, G. R. (2023), A machine learning-based approach for anomaly detection for secure cloud computing environments, *in* '2023 International Conference on IDCIoT'.

Parampottupadam, S. & Moldovann, A.-N. (2018), Cloud-based real-time network intrusion detection using deep learning, *in* '2018 International Conference on Cyber Security and Protection of Digital Services'.

Pitre, Gandhi, A., Konde, V. & Adhao, R. (2022), An intrusion detection system for zero-day attacks to reduce false positive rates, *in* '2022 International Conference ICONAT'.

Rabin, N. (2023), 'Unlocking the power of amazon cloudwatch anomaly detection for secrets manager', *Medium* .
URL: *https://medium.com/cyberark-engineering/unlocking-the-power-of-amazon-cloudwatch-anomaly-detection-for-secrets-manager-27a7ffd66498*

Ravindranathan, M. K., Sendil Vadivu, D. & Rajagopalan, N. (2024), Cloud-driven machine learning with aws: A comprehensive review of services, *in* '2024 International Conference on IITCEE'.

Shokrzad, R. (2023), '6 pivotal anomaly detection methods: From foundations to 2023's best practices', *Medium* .
URL: *https://medium.com/@reza.shokrzad/6-pivotal-anomaly-detection-methods-from-foundations-to-2023s-best-practices-5f037b530ae6*

Suresh babu, C., Surendar, V., Sriram, E. & Subhash, S. (2024), Web-based deep learning model for zero day vulnerability detection using fastapi, *in* '2024 International Conference on ADICS'.

Thudumu, S., Branch, P. & Jin, J. (2020), 'A comprehensive survey of anomaly detection techniques for high dimensional big data', *Journal of Big Data* **7**(1).
URL: *https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00320-x*

Touré, A., Imine, Y., Semnont, A., Delot, T. & Gallais, A. (2024), 'A framework for detecting zero-day exploits in network flows', *Computer Networks* .
URL: *https://www.sciencedirect.com/science/article/pii/S1389128624003086*

Zekri, M., Kafhali, S. E., Aboutabit, N. & Saadi, Y. (2017), Ddos attack detection using machine learning techniques in cloud computing environments, *in* '2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech)'.