Proceedings of 14th IOE Graduate Conference
Peer Reviewed
Year: 2023 Month: December Volume: 14
ISSN: 2350-8914 (Online), 2350-8906 (Print)

Analyzing Cloud Traffic for Web Honeypot using Microservice-based Architecture and XGBoost algorithm

Susmita Shrestha [a], Subarna Shakya [b], Anku Jaiswal [c]

[a, b, c] *Department of Electronics and Computer Engineering, Pulchowk Campus, IOE, Tribhuwan University, Nepal*
[a] 078mscsk019.susmita@pcampus.edu.np, [b] drss@ioe.edu.np, [c] anku.jaiswal@pcampus.edu.np

Abstract
The intricacy of web applications and the cloud computing industry's exponential expansion have created new difficulties for security. Web honeypots have become effective resources for identifying and examining harmful activity aimed at web applications. This paper suggests a novel approach for designing and analyzing web honeypots that makes use of a microservices architecture and machine learning methods. This proposed system leverages microservices-based architecture to enhance the scalability, flexibility, and modularity of the web honeypot infrastructure. By breaking down the traditional monolithic architecture into smaller, independent services, proposed method achieves better resource utilization and the ability to scale components individually based on the traffic load. This enables to effectively handle the high volume of cloud traffic, reducing latency and enhancing overall system performance. The study employs containerization methodologies with docker and kubernetes to deploy microservices in the cloud. Additionally, this research explores XGBoost machine learning algorithm into the honeypot system to analyze incoming traffic. Proposed system equips the honeypot with the ability to precisely recognize and categorize numerous sorts of attacks, including as SQL injection, cross-site scripting, and brute-force efforts by training the algorithms on large amounts of datasets of both legitimate and malicious traffic patterns.

Keywords
brute-force efforts, cloud computing, cross-site scripting, docker, kubernetes, machine learning, microservices architecture, SQL injection, web application, web honeypot and XGBoost.

1. Introduction

The rapid growth of cloud computing has caused a change in how organizations deploy and handle their web applications in recent years. With the popularity of cloud-based services increasing, cybersecurity vulnerabilities affecting cloud infrastructure and applications have also increased. Enterprises must have effective security systems that can recognize and evaluate dangerous activities in real-time in order to lower these risks.

The main idea behind microservices is a modular design that divides large applications into loosely tied autonomous components. A significantly greater number of independent components and the communication pathways connecting them make up the modular design of the microservices architecture. A system's attack surface grows as there are more components since there are more moving parts overall. As a result, protecting these systems becomes significantly more challenging and important. As technology advances, cyber-attacks are also evolving to exploit vulnerabilities and gain access to users' confidential information. Hacking attempts including phishing, crypto trojans, and cyber scamming are frequent and dangerous.

Using web honeypots, which are decoy systems designed to record cloud activity indicative of potential cyber-attacks. The traditional method for deploying honeypots was on physical or virtual computers, but with the introduction of machine learning techniques and microservices architecture, a new method has evolved. The research paper suggests use of the modular architectural design of microservices to build a distributed honeypot infrastructure in a cloud setting. The honeypot system becomes extremely scalable, fault-tolerant, and adaptive to handle enormous amounts of incoming traffic by utilizing microservices. Furthermore, the system employs machine learning techniques to identify and analyze the obtained traffic, distinguishing between legitimate and lethal user activities.

1.1 Cloud Computing

Cloud computing has emerged as a transformative technology that revolutionizes the way organizations deploy, manage, and utilize their IT resources. It provides an adaptable and expandable architecture for providing computer services through the internet, allowing enterprises to instantly access a huge variety of resources and applications. Cloud computing has fundamentally hindered the conventional paradigm of developing and maintaining IT infrastructure. Cloud computing allows for remote infrastructure to be applied based on needs.

1.2 Microservice-based Web Honeypot

A microservice-based web honeypot combines the principles of microservices architecture with the capabilities of web honeypots. Instead of using a monolithic approach, where the entire honeypot is a single system, it leverages the benefits of microservices to create a more flexible, scalable, and resilient infrastructure for hosting and managing multiple honeypot services. In this context, each microservice within the honeypot ecosystem represents a specific component or

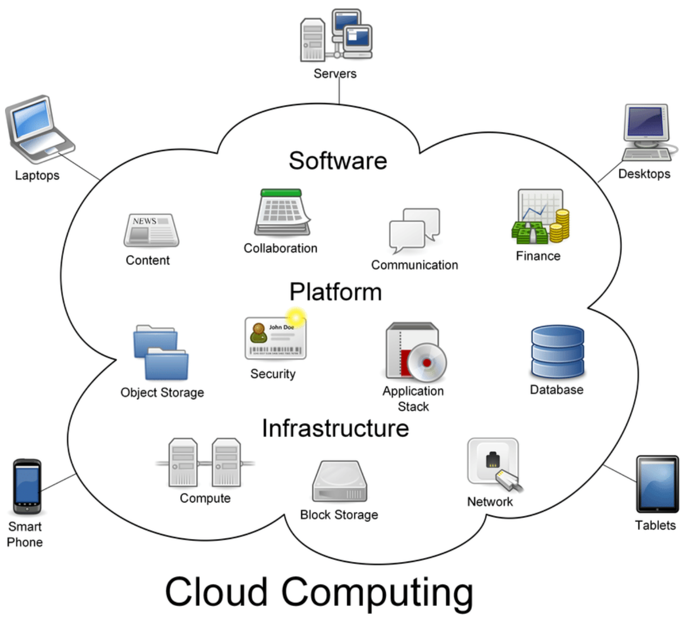


Figure 1: Cloud Computing Overview [1]

functionality of the overall honeypot system. For example, there might be microservices for authenticating users, capturing data, logging and monitoring activities. These microservices can be independently deployed, scaled, and updated, allowing security professionals to customize their honeypot infrastructure according to their specific needs and requirements

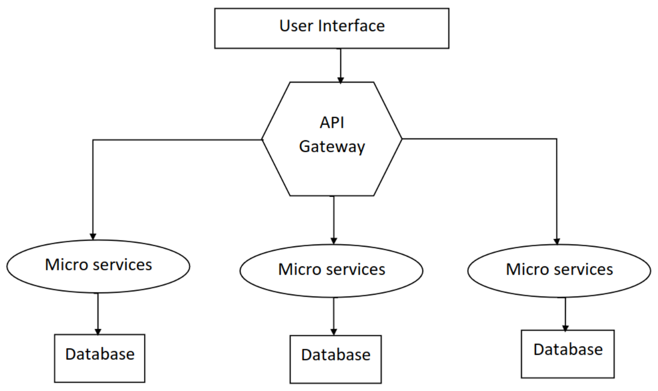


Figure 2: Microservice Overview

1.3 Machine Learning Algorithm

Extreme Gradient Boosting (XGBoost) is an advanced implementation of the gradient boosting machine learning technique. It is designed to optimize performance and speed by utilizing parallel computing and incorporating regularization techniques. XGBoost is particularly effective in solving complex regression and classification problems and has gained popularity for its outstanding performance in various data science competitions and real-world applications. It is an optimized implementation of the gradient boosting framework, which combines multiple weak prediction models i.e. decision trees to create a strong predictive model.

1.3.1 XGBoost Overview

Gradient Boosting framework aids to repeatedly construct an ensemble of weak models and combine their predictions to produce a stronger model. By progressively adding models that predict the residual errors of the prior models, it minimizes a given loss function. XGBoost uses regularization techniques to avoid overfitting and improve generalization, handles missing values, which eliminates the need for preprocessing processes, and is highly applicable for parallel processing and scalability.

2. Literature Review

The development of container orchestration platforms like Kubernetes (K8s) and microservices architecture has completely transformed the industry by enhancing scalability, flexibility, and deployment simplicity. Researchers are investigating these areas to identify vulnerabilities, develop best practices, and propose security solutions to mitigate risks.

Software defined networking architecture based honeynet is proposed a flexible and programmable network environment along with enhanced control of the network by separating the control plane from the data plane [2]. HoneyMix keeps a map of all available services in the network and generates data control rules in a centralized manner.

HoneyPLC presumably seeks to lure potential attackers and trick them into engaging with it by successfully impersonating actual devices, giving researchers or security experts a chance to examine their tactics, purposes, and potential weaknesses. Numerous commonly used reconnaissance programs, including Nmap, Shodan's Honeyscore, Siemens Step7 Manager, PLCinject, and PLCScan, have a high degree of confidence in HoneyPLC as genuine devices. Network scanning, device identification, vulnerability analysis[3].

The honeypot-as-a-service (HaaS) strategy aims to strike a balance between the effectiveness of honeypots against knowledgeable attackers and the economic considerations for businesses. It combines the advantages of honeypots in gathering valuable threat intelligence with the convenience and cost-effectiveness of a managed service[4]. The use of honeypots for intrusion detection and prevention has allowed users to exchange files by submitting requests, which are then examined depending on the credentials supplied to access the files[5].

On the Kubernetes orchestration platform, a technology called KubAnomaly offers security monitoring features for anomaly detection. A container monitoring module for Kubernetes is then developed, and neural network techniques are used to generate classification models that improve the system's capacity to detect unusual activities such web service attacks and common vulnerabilities and demonstrates attacks[6].

Adoption of cloud-native and microservices architectures raises new security concerns and emphasizes the necessity for specific security solutions like sandboxes. In this context, a "cloud-native sandbox" is an isolated, controlled environment where software or services may be used, tracked, and tested without disrupting the live system. It enables the study of potentially harmful actions and the discovery of holes or vulnerabilities inside the microservices ecosystem[7].

Several studies have been done which aims to explore the research in the field of intrusion detection, with a focus on the proposed method that emphasize the XGBoost algorithm. A approach for assessing the various data qualities in a network, such as precision, accuracy, and confusion matrix, is introduced in the research. The NSL-KDD dataset is used with XGBoost to learn more about data integrity, enhance the predictive capacity of data, and lessen the amount of hazardous data that is floating about in a network, creating a secure environment for exchanging information [8].

Extreme Gradient Boosting (XGBoost) is the basic foundation of the proposed approach, and the WOA (Whale Optimization Algorithm) is used to determine the optimal parameters for it[9].

Beyond behavioral analysis in security, XCBoost has been used in various domains. XGBoost has been used for prediction tasks including energy consumption, interior comfort, occupancy forecasting, or cost estimate in the early design stages of office buildings. The XGBoost model can uncover patterns and correlations between different design parameters and performance measures by being trained on past statistics or simulation results. In order to help architects and engineers make well-informed decisions throughout the design process, the trained model may then be used to forecast the performance of novel design configurations[10].

Containers are small, separated environments made possible by containerization technologies, such as Docker. The consistent and portable execution environment that containers offer for applications and services makes them ideal for setting up and maintaining honeypot systems. By utilizing the containerization capabilities of contemporary systems, the researchers seek to construct a honeypot solution that can be easily installed and run within networks or even outside of private networks in real-world situations[11].

The study, Containerized cloud-based honeypot deception for tracking attackers [12], provides a thorough overview of the implementation of containerized honeypots; as a result, they are transportable, resilient, and straightforward to deploy and administer. The instrumented method was tracked and produced a ton of data points from which it was possible to draw important conclusions about the actions and intentions of the malicious users.

The paper, A Cloud-Native Honeynet Automation and Orchestration Framework [13], proposes simulating attacker isolation on cloud-native systems with docker, integrating a virtual honeypot OS (Cowrie) to capture logs, and then utilizing threat modeling on the suggested architecture to qualitatively assess and provide a general framework for its implementation. It also addresses moving target defenses, which involve continually changing the configuration of the underlying system, making the honeynet implementation more difficult to understand.

The effectiveness of XGBoost and Catboost: tree-based classifiers in identifying the phishing websites has been studied. In terms of accuracy, both classifiers performed admirably. The outcome in particular demonstrates that XGBoost outperforms Catboost. These classifiers were examined using two datasets. K-fold validation and train-test validation are both employed during the study to support the result[14]. It is becoming common for fraudulent websites to infect users' devices. Users frequently browse these websites without paying attention to the URL details, which results in the theft of private data. Since attackers try to copy real URLs, it might be difficult to identify them.

The groundwater quality, which is jeopardized by excessive fuoride contamination, is also monitored using the ensemble machine learning technique. Light gradient boosting (LightGBM), random forest (RF), and extreme gradient boosting (Xgboost) were also employed in comparison with Hybrid Random Forest with a Linear Model for predicting fuoride contamination in groundwater. The accuracy of the xgboost model has been outstanding[15]. In contrast to the AdaBoost method, LightGBM and XGBoost both estimate the semantic textual similarity between the meaning of two texts and produce marginally superior results although LightGBM often performs somewhat better than XGBoost[16].

Boosting algorithms can be helpful in sectors like the food industry, where handling consumer meal orders has emerged as the key challenge. The proposed research uses gradient boosting regression models like Gradient Boosting, XGBoosting, LightGBM, CatBoost along with lasso, ridge, Bayesian ridge regression, Support Vector Regression, decision tree, and random forest to prevent inaccurate estimation of the food orders and waste of both food and raw materials, as well as ineffective employee management and decreased business profit[17].

The research paper, HoneyKube: Designing and Deploying a Microservices-based Web Honeypot[18], has explored creating and implementing a Microservices-based Web Honeypot for recording and analyzing network traffic, as well as logging and monitoring activities. Here, researchers introduce HoneyKube, a microservices-based web honeypot system that is both efficient and scalable. By adopting this approach, HoneyKube aims to provide accurate emulation of web services, gather threat intelligence, and enhance the understanding of web-based attacks and vulnerabilities.

3. Methodology

In order to construct a microservice based honeypot in a cloud environment, the problem domain should be defined and assessed at first. Following that, we take a two-step approach: first, we create a real-world application using the microservice architecture to authenticate users and gather service requests from users, and then we utilize the XGBoost algorithm to analyze and classify the stored request data.

3.1 Proposed System

3.2 XGBoost Algorithm

The detailed stepwise procedure for using the XGBoost algorithm for classification is given as:

- Initial Prediction: The Probability For The Base Model is set to P.

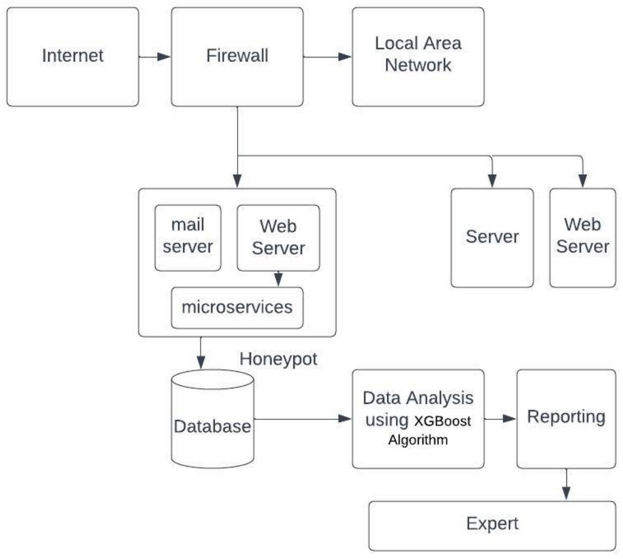


Figure 3: Block Diagram

- Similarity Score and Gain: It is given as:

$$SimilarityScore = \frac{\sum Residual^2}{\sum[P*(1-P)]+\lambda}$$

$$Gain = Left_{SS} + Right_{SS} - Root_{SS}$$

  Where P is Probability.
  lambda is the regularization parameter, which helps prevent overfitting.

- Prune the Tree: The tree is pruned based on the calculated gain and the selected gamma value.
  If Gain - Gamma is greater than 0, Keep the tree.
  If Gain - Gamma is less than 0, Prune the tree.

$$OutputValue = \frac{\sum Residual}{\sum[P*(1-P)]+\lambda}$$

- Predicted Value:

$$Predicted\ Value = Initial\ prediction + eta(learning\ rate) * output$$

4. Experimental Setup

4.1 Hardware and Infrastructure

The experimental setup aimed to assess the performance, scalability, and reliability of microservices orchestrated using Kubernetes and Docker. The following hardware and infrastructure components were utilized:

- Cluster Configuration: A cluster consisting of n nodes was set up. These nodes were dedicated to running Docker containers orchestrated by Kubernetes. The cluster was created in order to simulate a real-world cloud environment.
- Node Specifications: Each node in the cluster had two CPUs and 400 mi of storage. The hardware was chosen to reflect a cloud computing environment with dynamic resource allocation.
- Networking: Nodes were interconnected with each other over a local network created by Docker or the container runtime.

4.2 Software Stack

The software stack for our experimental setup comprised the following components:

- Operating System: All nodes ran a compatible Linux distribution, specifically 22.04 version.
- Containerization: Docker version 24.0.5 was installed on each node to create and manage containers for our microservices. Docker images for individual microservices were built according to predefined Dockerfiles.
- Orchestration: Kubernetes (Client Version: v1.28.1, Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3, Server Version: v1.27.4) was deployed on the cluster to orchestrate the deployment, scaling, and management of microservices. Kubernetes was configured to manage microservices defined in Kubernetes manifests.
- Monitoring and Logging: We employed monitoring tools Prometheus and Grafana for collecting data from containers.
- Programming language: We designed and developed the microservices in Python 3.11.5. We applied mysql (Ver 8.0.34-0ubuntu0.22.04.1) as the database system for user authentication. MongoDB was also used to serve the other services. To avoid request overflow, the message broker Rabbitmq is utilized to send the request in queue.

4.3 Microservices Deployment

This research was focused on a set of microservices representing a simplified video to mp3 converter application. These microservices included but were not limited to:

- Auth Service
- User Service
- Validate Service
- Gateway Service
- Convertor Service
- Download Service

Each microservice was containerized using Docker and encapsulated its dependencies, configurations, and runtime environment.

4.4 Kubernetes Manifests

For each microservice, we defined Kubernetes manifests in YAML format. These manifests included specifications for Deployments, Services, ConfigMaps, and Resource Requests and Limits.

4.5 Experiment Execution

The experiments were designed to evaluate various aspects of the orchestrated microservices, including storing and analyzing data request pattern from user and potential gaps in microservice architecture in a cloud context.

4.6 Data Collection and Analysis

Data collected during the experiment logs were recorded and analyzed using XGBoost machine learning algorithm.

5. Result and Analysis

This research aims to examine attack patterns in micro-service architectures by utilizing network-collected datasets. The input characteristics of the datasets includes source point details, destination point details, IP packets detail, login details, type of intrusion in the cloud for inspection of the micro-service application orchestrated in kubernetes platform.

5.1 Data Preprocessing and Feature Analysis

The datasets including various types of intrusions are the datapoints utilized to train the algorithm, as shown in figure 4.

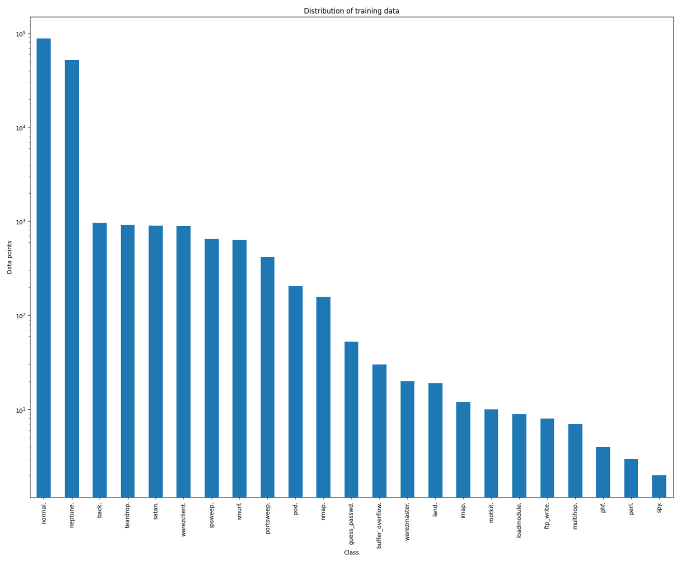


Figure 4: Intrusion Types

In the data preprocessing and exploratory data analysis stages, feature correlations have been examined. This analysis aids in the decision-making process when selecting features, as shown in figure 5.

5.2 Model Training and Analysis

In this research study, experiments were conducted to assess the accuracy of the research model, with a comprehensive examination of its performance across various samples. The accuracy plot, shown in Figure below, depicts the model's performance on the dataset. While assessing the model, the result of varying parameter values in the model is plotted to

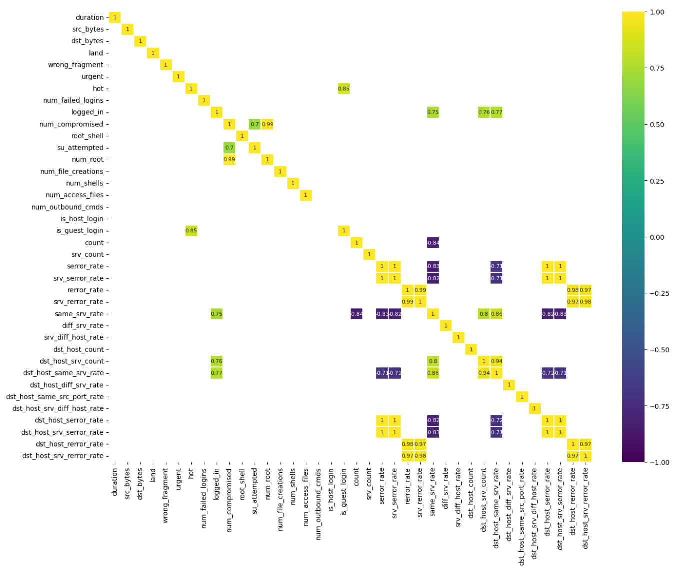
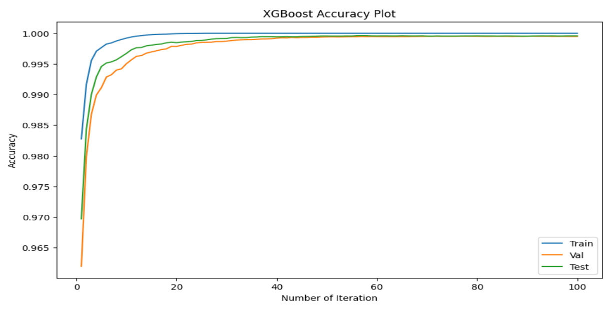
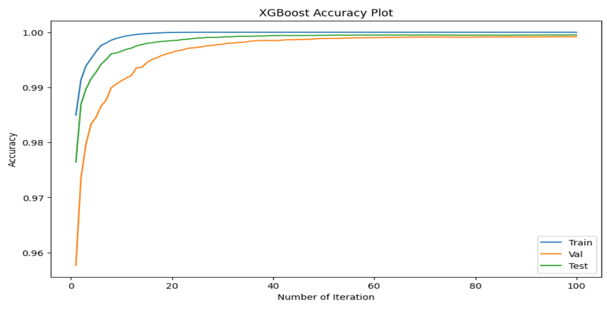


Figure 5: Features Correlation

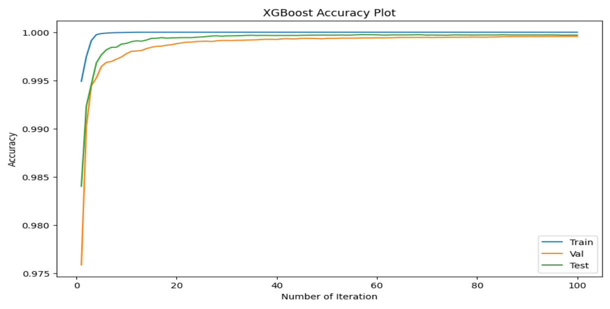
visualize the output in the form of accuracy. The accuracy plot is determined by adjusting hyperparameter values such as learning rate, depth of the tree in model, and sub-sample datasets used to train the tree in the model.



(a) Accuracy at Learning rate 0.2, depth 3 and sub-sample 0.8



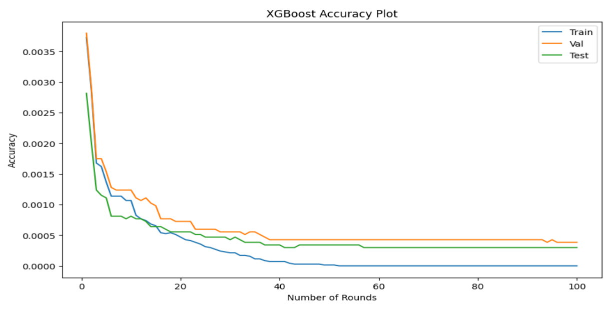
(b) Accuracy at Learning rate 0.2, depth 3 and sub-sample 1



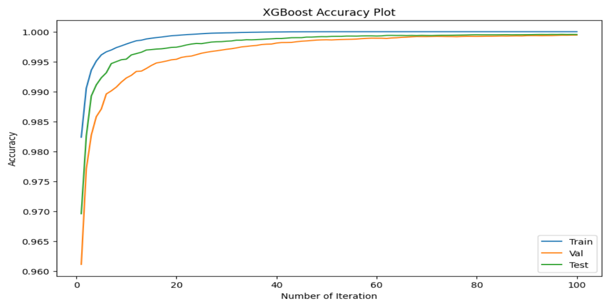
(c) Accuracy at Learning rate 0.2, depth 5 and sub-sample 0.8

Figure 6: Accuracy at different learning rates, depth, and sub samples

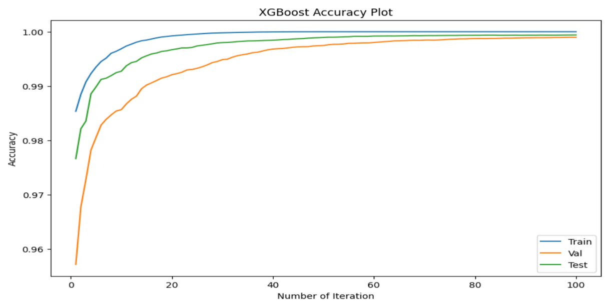
Figures 6a, 6b, 6c, 7a, 7b, 7c and 7d illustrate the varied accuracy plots with varying values of the hyper parameters. Figures 8a, 8b, 8c and 8d illustrate the varied loss plots with varying values of the hyper parameters.



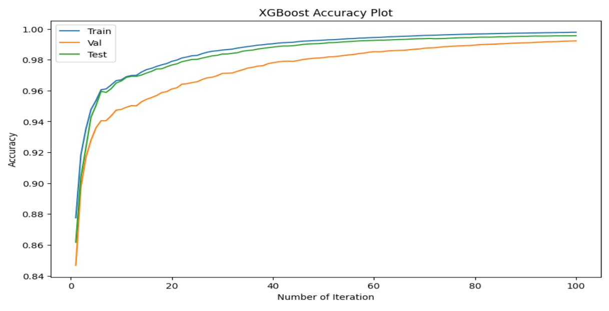
(a) Accuracy at Learning rate 0.2, depth 5 and sub-sample 1



(b) Accuracy at Learning rate 0.1, depth 3 and sub-sample 0.8



(c) Accuracy at Learning rate 0.1, depth 3 and sub-sample 1

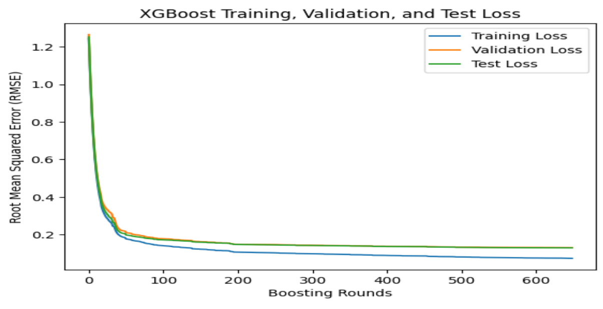


(d) Accuracy at Learning rate 0.1, depth 1 and sub-sample 0.8

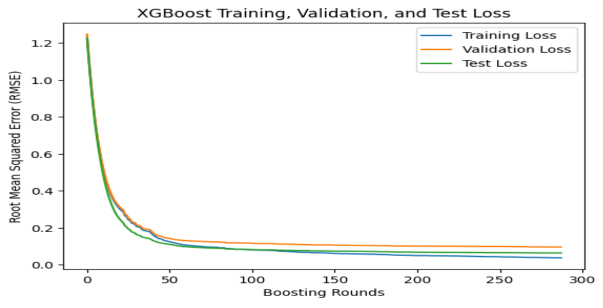
Figure 7: Accuracy at different learning rates, depth, and sub samples

The accuracy plot provides valuable insights into the model's performance under different hyperparameter configurations. Notably, as the depth of the tree increases, there is a discernible trend towards overfitting, evidenced by a decrease in performance. Conversely, when the tree depth is limited, the model demonstrates robustness and generalizability.

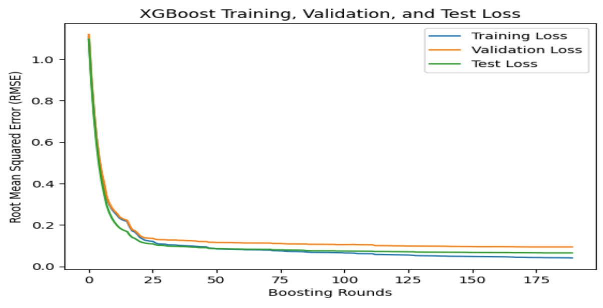
Additionally, the sub-sample value appears to significantly impact model performance during testing. A lower sub-sample value corresponds to enhanced performance, indicating that



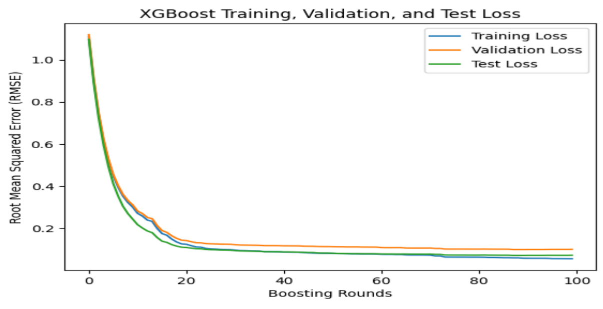
(a) Loss at Learning rate 0.1, depth 3 and sub-sample 0.8



(b) Loss at Learning rate 0.1, depth 5 and sub-sample 1



(c) Loss at Learning rate 0.2, depth 5 and sub-sample 1



(d) Loss at Learning rate 0.2, depth 5 and sub-sample 1

Figure 8: Loss at different learning rates, depth, and sub samples

the model benefits from a more restrained sampling approach.

Examining the impact of learning rate on accuracy reveals interesting dynamics. Specifically, learning rates of 0.1 and 0.2 introduce variability in accuracy, with nuanced fluctuations observed. Notably, a learning rate of 0.1 appears to yield more consistent and favorable results.

In summary, after careful analysis, optimal hyperparameter settings for this model emerge. The model performs optimally with a learning rate of 0.1, a tree depth of 1, and a sub-sample value of 0.8. These configurations collectively contribute to a well-balanced model that avoids overfitting, maintains robustness, and exhibits superior performance in testing.

5.3 Comparison of Model Performance with Intrusion Type

The table 1 depicts the findings of a model's performance as measured by evaluation metrics. These metrics have significance for determining a model's efficacy in data analysis, machine learning, and classification.

Table 1: Output Status for Xgboost

S.N.	Evaluation Metrics	Value
1	Accuracy	0.999378109
2	Precision	0.999344343
3	Recall	0.999378109
4	F1-score	0.99931787

Table 2 provides a thorough summary of a XGBoost model's performance when compared to various intrusion types. It uses assessment metrics to show the effectiveness of the model.

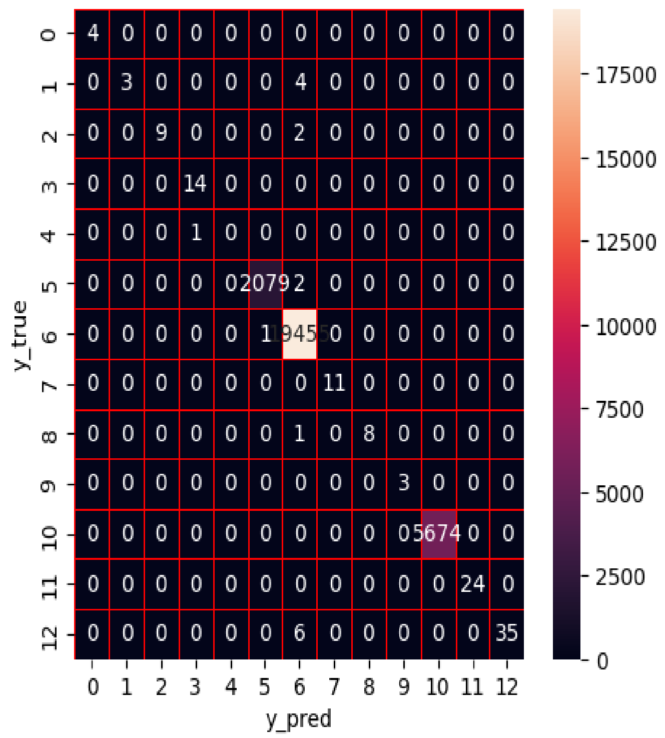
A graphical representation of confusion metrics associated with an XGBoost model's performance is shown in Figure 9. Based on the model's predictions for a given dataset, these metrics are calculated.

The table 2 displays model performance across various intrusion types as shown by various evaluation metrics. This variation emphasizes the importance of intrusion in model performance. It implies that the model's effectiveness is determined by the precise nature of the incursion being addressed, revealing the intricacies of its predictive skills across various circumstances. This variation could be due to differences in dataset sizes for each intrusion type. Notably, fluctuations in accuracy indicate a proclivity for overfitting during the model's training phase, which have impaired its capacity to generalize effectively to different intrusion circumstances. Although there appears to be minimal overfitting in the training phase, the model works very effectively on the analysis of network data.

Table 2: Output Status for Various Intrusion Types

S.N.	Types of Intrusion	precision	recall	f1-score	support
0	back	1.00	1.00	1.00	4
1	buffer overflow	1.00	0.43	0.60	7
2	guess passwd	1.00	0.82	0.90	11
3	ipsweep	0.93	1.00	0.97	14
4	land	0.00	0.00	0.00	1
5	neptune	1.00	1.00	1.00	2081
6	normal	1.00	1.00	1.00	19456
7	pod	1.00	1.00	1.00	11
8	portsweep	1.00	0.89	0.94	9
9	satan	1.00	1.00	1.00	3
10	smurf	1.00	1.00	1.00	5674
11	teardrop	1.00	1.00	1.00	24
12	warezclient	1.00	0.85	0.92	41

We can observe diverse intrusion types during the experiment. Exploitation of various types of application vulnerabilities can contribute to a variety of attacks, such as Smurf, Neptune, Warez client, Portscan (e.g., portsweep, ipsweep), Guess



$$
\begin{array}{c|ccccccccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\hline
0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 3 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 9 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & 0 & 2079 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 0 & 0 & 0 & 0 & 0 & 1 & 19455 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 & 0 & 0 & 0 & 0 & 0 \\
8 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 8 & 0 & 0 & 0 & 0 \\
9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\
10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5674 & 0 & 0 \\
11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 24 & 0 \\
12 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 & 35
\end{array}
$$

Figure 9: Confusion Metrics for Xgboost Model

Password, Teardrop, Buffer Overflow, Backdoor (Back), Satan, and Land attacks. Inadequate network segmentation can cause it easier for attackers to execute scanning and reconnaissance activities, leading to attacks such as Smurf, portsweep, and ipsweep. Lack of encryption and secure communication protocols can expose microservices to sniffing attacks and spoofing attempts, contributing to attacks like the Land attack. Attackers can employ exposed or unprotected APIs to introduce backdoors, assist the distribution of malicious software (Warez), or do other unauthorized acts. Unauthorized access can be caused by weak or misconfigured authentication methods.

To obtain access, attackers can opt for brute force assaults (Guess Password) or take advantage of weak passwords. Lack of proper input validation in microservices can allow attackers to inject malicious code, leading to buffer overflow vulnerabilities. Using outdated or vulnerable third-party libraries or components in microservices can expose the system to known vulnerabilities, as seen in the Neptune attack. Exposed management interfaces with weak authentication can be targeted for unauthorized access and exploitation, contributing to attacks like Guess Password or Backdoor.

Inadequate logging and monitoring procedures can make it difficult to recognize and respond to various kinds of attacks, such as malicious activity, illegal access, and reconnaissance efforts like Ipsweep and Portscan. Attackers may be able to carry out a number of assaults, due to misconfigurations in cloud services, such as unsafe storage or network configurations. Microservices are susceptible to network-based assaults such as Teardrop and Land attacks if they have inadequate network security security in place, such as weak firewalls or incorrectly configured security groups.

Therefore, conducting experiments like this enables the monitoring of potential invasions in the cloud through regular assessments, proactive monitoring, audits, and threat awareness.

6. Comparison with Existing Works

The research paper HoneyKube[18] focuses on the development and deployment of a Microservices-based Web Honeypot. It likely delves into how HoneyKube enhances cybersecurity through the simulation of web services to detect and analyze potential threats. It lacks the through analysis of the datasets collected during the simulation which facilitates in predicting potential future disasters caused by the microservice design flaw, which finally hampers and exposes credible data to unauthorized entities. While it has gathered system traces, network traces, and Kubernetes audit logs, as well as introduces vulnerabilities to services, it faces limitations in effectively analyzing various types of attacks and establishing the root causes of these attacks. The system falls short in providing comprehensive insights into the nature and origin of different attacks, potentially limiting its ability to offer a thorough understanding of the security threats it encounters. By building a containerized cloud-based honeypot system, the research study [12] provides a unique method to cybersecurity. This system functions as a deception technique, luring and tracking intruders. Despite the fact that the research mainly focused on DDoS attacks, it also fails to provide an early warning when DDoS traffic is detected.

Table 3: Comparison between this work and others

Performance Metrics	[19]	[20]	[21]	This Work
Accuracy	0.97	0.994	0.967	0.999378
Precision	0.99	0.994	0.987	0.999344
Recall	0.98	0.994	0.943	0.999378
F1-score	0.97	0.994	0.964	0.999317

This research aims to assess data collected through the use of the XGBoost algorithm, aiming to identify attack patterns and various attack variations within the network. Additionally, the study introduces a approach called request queuing, facilitating the monitoring of potential attacks and contributing to the prevention of Distributed Denial of Service (DDoS) attacks to a certain extent. In this thesis, the utilization of the XGBoost algorithm stands out for its efficiency in detecting and analyzing gathered threats with high accuracy and minimal training time. The comparison is conducted by evaluating the performance of various models employed in prior research papers alongside the current work. Upon examining the accuracy metrics, it is evident that the model's performance in this work surpasses that of the models used in previous research. In particular, the accuracy values reported in Papers [19, 20, 21] are 0.97, 0.994, and 0.967, respectively. In contrast, the accuracy achieved in this thesis work outperforms all, with a notable accuracy of 0.9993. The application of this model has enabled the research to effectively analyze diverse techniques employed by attackers, showcasing the algorithm's capability to provide precise insights into various threat scenarios. The efficiency of XGBoost contributes to a thorough understanding of potential security risks, enhancing the overall effectiveness of threat detection and analysis in this research.

7. Limitation and Future Enhancement

A system compromise to the external environment during the data collection process increases the risk of further security breaches. Thus, the system must be carefully and thoroughly monitored. This research focused solely at the XGBoost algorithm in this study in order to analyze attack patterns; therefore, a hybrid algorithm would be preferable for analyzing data points. Furthermore, since the cloud expands rapidly, researchers may consider introducing additional constraints to address cloud security that aren't limited to specific platforms.

8. Conclusion

This research has significant contribution to the field of security. The integration of a cloud-based microservice honeypot with machine learning algorithms provides a comprehensive and proactive approach to identifying security risks. This study involves design and deployment of a microservice-based honeypot for simulating and monitoring potential security threats. Building upon this foundation, the XGBoost algorithm was then applied to analyze the datasets that have been collected during the simulation. By leveraging machine learning techniques, this research systematically processed the collected data, enabling the identification and categorization of intricate attack patterns. This analytical approach not only enhances understanding of evolving cyber threats but also contributes to the development of more effective and adaptive cybersecurity strategies.

Acknowledgments

The authors extend their sincere gratitude to all individuals and Department of Electronics and Computer Engineering, IOE, Pulchowk Campus whose contributions were essential in the completion of this research. We appreciate the invaluable guidance, support, and resources provided by our mentors, colleagues, and academic advisors throughout the research process. Special thanks go to the researcher who provided their time and insights for this research.

References

[1] M. Duggan. The application of machine learning to optimise live migration in cloud data centres. *ResearchGate*, 2019.

[2] W. Han, Z. Zhao, A. Doupé, and G. J. Ahn. Honeymix: Toward sdn-based intelligent honeynet. In *In Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, pages 1–6, 2016.

[3] E. López-Morales, C. Rubio-Medrano, A. Doupé, Y. Shoshitaishvili, R. Wang, T. Bao, and G. J. Ahn. Honeyplc: A next-generation honeypot for industrial control systems. *In Proceedings of the 2020 ACM SIGSAC*

Conference on Computer and Communications Security, pages 279–291, 2020.

[4] J. H. Jafarian and A. Niakanlahiji. Delivering honeypots as a service. *In HICSS*, pages 1–10, 2020.

[5] P. S. Negi, A. Garg, and R. Lal. Intrusion detection and prevention using honeypot network for cloud security. In *In 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 129–132, 2020.

[6] C. W. Tien, T. Y. Huang, C. W. Tien, T. C. Huang, and S. Y. Kuo. Kubanomaly: anomaly detection for the docker orchestration platform with neural network approaches. *Engineering reports*, 2019.

[7] Z. Xu and T. Luo. Cloud-native sandboxes for microservices: Understanding new threats and attacks. In *Blackhat Europe*. 2018.

[8] S. S. Dhaliwal, A. A. Nahid, and R. Abbas. Effective intrusion detection system using xgboost. In *Information, 9(7)*, 2018.

[9] Y. Song, H. Li, P. Xu, and D. Liu. A method of intrusion detection based on woa-xgboost algorithm. In *Discrete Dynamics in Nature and Society*, 2022.

[10] H. Yan, K. Yan, and G. Ji. Optimization and prediction in the early design stage of office buildings using genetic and xgboost algorithms. In *Building and Environment, 218, 109081.*, 2022.

[11] G. Purswani, N. Hemant, R. K. Singh, and R. Parashar. Honeypot in a container with detailed analytics. In *Applied and Computational Engineering.*, pages 327–335, 2023.

[12] VS Devi Priya and S Sibi Chakkaravarthy. Containerized cloud-based honeypot deception for tracking attackers. *Scientific Reports*, 13(1):1437, 2023.

[13] Akash Ravi, Bhavye Sharma, and Avigyan Mukherjee. A cloud-native honeynet automation and orchestration framework.

[14] K. Sadaf. Phishing website detection using xgboost and catboost classifiers. In *In 2023 International Conference on Smart Computing and Application (ICSCA)*, pages 1–6, 2023.

[15] Mouigni Baraka Nafouanti, Junxia Li, Edwin E Nyakilla, Grant Charles Mwakipunda, and Alvin Mulashani. A novel hybrid random forest linear model approach for forecasting groundwater fluoride contamination. *Environmental Science and Pollution Research*, 30(17):50661–50674, 2023.

[16] Ivan Rep and Vladimir Čeperić. Boosting the performance of transformer architectures for semantic textual similarity. *arXiv preprint arXiv:2306.00708*, 2023.

[17] Sasikumar Jayapal. *Food Demand Prediction using Statistical and Machine Learning Models*. PhD thesis, Dublin, National College of Ireland, 2023.

[18] C. Gupta, T. van Ede, and A. Continella. Honeykube: Designing and deploying a microservices-based web honeypot. In *SecWeb*, 2023.

[19] B. S. Sukhadeo, R. N. Patil, R. Atole, Y. D. Sinkar, U. C. Patkar, and R. Chopade. Mlids: A machine learning-based intrusion detection system using the nslkdd data. *International Journal of Intelligent Systems and Applications in Engineering*, 2023.

[20] D. Jeevaraj, B. Karthik, M. Sriram, S. P. Vijayaragavan, , and D. Gokulakrishnan. Intrusion detection in wsn using supervised machine learning techniques. *International Journal of Intelligent Systems and Applications in Engineering, 12(9s), pp. 483–490.*, 2023.

[21] Zahedi Azam, Md. Motaharul Islam, and Mohammad Nurul Huda. Comparative analysis of intrusion detection systems and machine learning-based model analysis through decision tree. *IEEE Access*, 11:80348–80391, 2023.