National University Of Computing & Emerging Sciences

SE-4091 Final Year Project Proposal

Fall 2024, BS-SE

Comparison of Deep Learning and Machine Learning in DoS Detection in Kubernetes

Team

Asher Siddque - 21K-3860

Mahnoor Amjad - 21K-3884

Rija Anwar - 21K-3069

Supervisor

Shoaib Raza (shoaib.raza@nu.edu.pk)

Department Of Software Engineering

National University Of Computing And Emerging Sciences

Date of delivery, *September 18, 2024*

Contents

Abstract

The build-out of Cloud native applications and micro-services architecture across various systems and tech in the industry has created for a dire need of an effective solution to Denial of Service and Distributed Denial of Service attacks. Providing such solutions is challenging due to the highly dynamic and distributed nature of micro-services. Having reviewed the classical Machine Learning models deployed in generalized anomaly detection in Kubernetes-orchestrated environment, along with the deployment of a single deep learning model (LSTM) for a more targeted approach towards detection, we aim to build upon the gap by introducing a hybrid approach for the cause of detection of DoS/DDoS attacks. The mentioned approach will compare chosen Machine Learning models' benchmarks with the Deep Learning models to decide upon the most optimal solution that not only provides real-time security but does so in a manner that reduces operational costs whilst maintaining high performance and progressing a cost and energy efficient production environment.

1 Introduction

1.1 Project Summary

We aim to create a system that can detect Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks that micro-services fall prey to in a Kubernetes environment. Micro-services are modular, loosely coupled services often deployed across multiple systems at once, the detection of said attacks becomes more complex. Our system will collect resource metrics from various micro-service nodes and use two Machine Learning models and two Deep Learning models to identify attack patterns. The primary goal is to compare the efficiency of each model in terms of accuracy, speed, resource consumption, and overall performance.

So far, only one type of Deep Learning model (LSTM with Autoencoders) has been deployed, alongside Machine Learning models designed for a more generalized concept of detecting anomalies based off on application and system logs. In this project, we aim to compare the domains of Machine Learning models against Deep Learning models and try to find an optimal solution for detecting attacks on cloud-native applications. The system will identify abnormal traffic patterns that signal the occurrence of these attacks, using real-time data comprised of system calls, and resource metrics taken in real time from Kubernetes. To enhance accuracy and reliability, we intend to farm our own data to generate independent and more precise sequences for comparison.

The Deep Learning models we shall be deploying are stated as below:

LSTM (Long Short-Term Memory): A type of RNN designed to capture long-term dependencies in sequential data by using memory cells to maintain information over time.

GRU (Gated Recurrent Unit): A simplified version of LSTM that uses gating mechanisms to efficiently capture short- and long-term dependencies in sequential data, requiring fewer resources.

The Machine Learning Models we shall be deploying are stated as below:

SVM (Support Vector Machines)

RF (Random Forests)

By comparing these models, the project aims to recommend the most effective solution for real-time DoS and DDoS attack detection in Kubernetes-based micro-service environments. This analysis is critical for improving security while maintaining system performance in cloud-native applications.

1.2 Project Objectives

1. To compare the chosen Deep Learning models in the regard of detecting DoS and DDoS attacks and figure out the best model from the comparison.
2. To compare the chosen Machine Learning in the regard of detecting DoS and DDoS attacks and figure out the best model from the comparison.
3. To compare the final Machine Learning and Deep Learning models in the regard of detecting DoS and DDoS attacks and figure out the most optimal solution from the comparison.

4. To develop a real-time monitoring and detection system for identifying abnormal traffic patterns indicative of DoS and DDoS attacks in a Kubernetes micro-services environment.

5. To evaluate the accuracy, speed, and resource efficiency of each Deep Learning model in detecting network-based attacks, ensuring minimal impact on system performance,

6. To implement and refine the novel agent service from the base paper to collect real-time metrics from Kubernetes nodes/pods, creating a dataset for training the Deep Learning models for specific DoS and DDoS attack detection.

2 Overview

2.1 Literature Review

Rapid growth of micro-service architecture and Kubernetes has brought about an intensity in the need for a set of effective mechanisms to deal with DoS and DDoS attacks in cloud native environments. Our review spans over two key research areas; Classical machine learning for general anomaly detection in micro-services and containers, and deep learning approaches specifically targeting DoS/DDoS detection in Kubernetes environments.

Classical machine learning techniques have been explored and deployed for anomaly detection in cloud and containerized environments. Lin et al.[1] proposed a Classified Distributed Learning system, utilizing random forests and unsupervised neural networks for detecting anomalies in container applications. Darwesh et al.[2] addressed data collection challenges in Kubernetes with an agent-based approach, while Forsberg[3] introduced a clustering-based system for detecting abnormal patterns in micro-service clusters. Zou et al.[4] optimized the Isolation Forest algorithm for dynamic container environments, and Du et al.[5] compared multiple machine learning algorithms for anomaly detection using performance metrics. Anemogiannis[6] proposed a hierarchical detection system tailored for Kubernetes clusters, highlighting the platform's complexity.

Deep learning techniques, such as Long Short-Term Memory (LSTM) networks and Autoencoders, have been employed to specifically detect DoS/DDoS attacks. Tien et al.[7] developed KubAnomaly, combining system call monitoring with neural networks for detecting container anomalies. Harlicaj[8] presented two methods for detecting web-based attacks: one using LSTMs to monitor communication between micro-services and another employing Autoencoders to analyze web request anomalies. Both approaches show promise for deep learning in enhancing attack detection within containerized environments.

Despite such progress, challenges remain in terms of real-time detection, scalability, minimizing false positives, and adaptation of evolving attack patterns. From this we easily attain our respected research gap which is to explore a hybrid solution to our problem of detecting DoS/DDoS attacks in cloud native applications. The said approach would combine machine learning and deep learning and pose to aid in improving detection systems in production environments with low overhead.

2.2 Benefits of the Project

2.2.1 Industrial Benefits

Current systems may struggle with real-time detection, especially under high traffic loads. By leveraging Deep Learning models, this project enables faster and more adaptive and accurate detection, allowing developer teams to respond to these attacks attacks in real time with minimal delay. This reduces downtime and system vulnerability during an attack.

Traditional cloud native and cybersecurity detection systems can be resource-intensive, leading to high operational costs. By comparing multiple Machine Learning and Deep Learning models, this project identifies resource-efficient models (e.g., GRU), enabling security systems to consume less computational power while maintaining high performance and low system latency. This can significantly reduce overhead and create an energy and cost efficient environment for companies that deploy micro-services.

This research provides us with a scalable solution, as many systems often struggle with scaling their mechanisms as infrastructure grows. The solution is not merely integrated but is made specifically for dynamic environments like Kubernetes, ensuring that security keeps pace with expanding micro-service deployments without performance degradation.

Industries with critical infrastructure (e.g., finance, healthcare) can benefit from faster, real-time attack detection, minimizing the impact of cyberattacks on their operations.

2.2.2 Community Benefits

This research provides valuable insights into effective DoS and DDoS detection mechanisms, which can be leveraged in our open-source project focused on Kubernetes and micro-services platforms. By making our findings and implementations freely available, we aim to benefit developers and small businesses seeking to enhance their cybersecurity measures.

The comparative study of Deep Learning and Machine Learning models for micro-services and cybersecurity applications can be used as a reference for students, researchers, and practitioners in the field, contributing to academic advancement.

Community-based cloud services, platforms, and micro-service architectures can adopt these models to prevent detect large-scale disruptions from cyberattacks.

The research can inspire developers to build more secure and efficient micro-service architectures by implementing advanced attack detection systems.

3 Project Implementation Method

3.1 Finding Appropriate Research Gap

Our project began with a comprehensive background study in the line of DoS and DDoS attack detection. Majority of the studies focused on general anomaly detection in cloud environments, but we focused on three papers that specifically addressed DoS/DDoS detection in Kubernetes-based micro-services. From these works, we extracted relevant research gaps that will form the basis of our investigation throughout this project.

3.2 Data Collection

The implementation phase will begin with the development of a custom data collection script. Due to the scarcity of public datasets for DoS/DDoS attacks on Kubernetes systems, we will generate our own private dataset. Our approach involves:

1. Deploying a local Kubernetes cluster running an open-source micro-service application.
2. Developing a custom script to:
   (a) Simulate Normal Usage and DoS/DDoS attacks on the cluster.
   (b) Collect normal and attack data in real-time.
   (c) Save the collected data locally in an appropriate format for model training.

This method ensures we have a high-quality, relevant dataset tailored to our specific use case.

3.3 System Implementation

We will create an agent service utilizing Sysdig and/or Falco to:

1. Collect syscall data and usage metrics from the Kubernetes cluster.
2. Transmit this data to the Data Flow Aggregator.

The Data Flow Aggregator will

1. Handle incoming data.
2. Pass the data to the selected model.
3. Associate the model's output with the input data.
4. Store results in a database.

Additionally, we will set up a web server to serve a dashboard. This dashboard will asynchronously display real-time updates about the system's status and attack detection.

3.4 Model Implementation and Concurrent Testing

As outlined in the Project Objectives, we will implement and compare:

1. Two Deep Learning models
   (a) LSTM (Long Short-Term Memory)
   (b) GRU (Gated Recurrent Unit)
2. Two Machine Learning models
   (a) SVM (Support Vector Machines)
   (b) RF (Random Forests)

The development of these models will be complemented by rigorous testing using a portion of the dataset.

3.5 Testing on Real Time Streaming Data

We are adopting a modular approach to facilitate easy model swapping within the fully integrated system. This design allows us to

1. Test each developed model with real-time data streams.
2. Compare the performance of Machine Learning and Deep Learning models in a live environment.
3. Evaluate the models' effectiveness in detecting DoS and DDoS attacks on the deployed Kubernetes cluster.

By implementing and concurrently testing these models with real-time data, we aim to identify the most effective solution for DoS and DDoS attack detection in cloud-native applications, balancing accuracy, speed, and resource consumption.

3.6 Evaluation and Further Improvements

To ensure a comprehensive assessment of our DoS and DDoS detection system, we will implement rigorous evaluation protocols for both individual models and the overall system. We will focus on strategies to reduce end-to-end latency.

3.6.1 Model Evaluation and Comparison

We will evaluate and compare the performance of our Machine Learning (SVM, RF) and Deep Learning (LSTM, GRU) models using the following metrics

1. **Accuracy** Measure the overall correct classification rate.
2. **Precision and Recall** Assess the models' ability to correctly identify attacks while minimizing false positives.

3. **F1-Score** Provide a balanced measure of precision and recall.
4. **Detection Speed** Measure the time taken to classify incoming traffic patterns.
5. **Resource Utilization** Monitor CPU and memory usage during model inference.

This comprehensive evaluation will allow us to identify the most effective model for our specific use case in Kubernetes-based micro-services environments.

3.6.2 System Evaluation

To assess the overall performance of our detection system, we will

1. Conduct end-to-end testing using simulated DoS and DDoS attacks on our Kubernetes cluster.
2. Measure the system's response time from the moment an attack begins to its detection and alert generation.
3. Evaluate the system's scalability by gradually increasing the cluster size and traffic volume.
4. Assess the impact of our detection system on the performance of the monitored micro-services.

By focusing on these evaluation methods and latency optimization strategies, we aim to create a robust, efficient, and low-latency DoS and DDoS detection system tailored for Kubernetes-based micro-services environments. Our goal is to achieve real-time threat detection while minimizing the performance impact on the monitored systems.

4 Final Deliverables of the Project

As per the requirements for the semester, the deliverable of the FYP will include,

1. 30% prototype for FYP-I
2. 100% for FYP-II

4.1 FYP-I

On the documentation front, FYP-I will be submitting this "*FYP Project Proposal*" for the defense, and the "*Research Report*"

As for the product side, FYP-I will include the collection of the private dataset, implementation of system and development of the two Machine Learning models.

Overall, this phase includes working with the following:

1. Research Gap
2. Data Collection

3. Kubernetes Agent Service
4. Two Machine Learning models

4.2 FYP-II

FYP-II will move towards finalization of the FYP, as well as the delivery of the following objects

1. Codebase for the following
   (a) Kubernetes Agent Service
   (b) Web-server and Dashboard
   (c) Data Flow Aggregator
2. Full Working System
3. Machine Learning and Deep Learning Models
4. Dataset
5. Research Paper
6. Literature Review Report

5 Technical Details of the Final Deliverable

Our system architecture consists of three main components: an agent-based service, a central module for handling models and NoSQL database management, and a web-server that provides a dashboard for the development team. This design ensures efficient data collection, processing, and visualization for DoS and DDoS attack detection in Kubernetes environments.

5.1 Kubernetes Agent Service

The agent service is deployed within the Kubernetes cluster and comprises agents developed using Sysdig and Falco. Key features include

1. Per-node deployment to ensure comprehensive coverage of the cluster.
2. Collection of system calls and resource metrics from each node and pod.
3. Data transmission to the Data Flow Aggregator at 10-second intervals, balancing between timely updates and minimal cluster overhead.
4. Built upon the foundation of our base paper's methodology[cite here], adapted specifically for DoS and DDoS detection.

5.2 Data Flow Aggregator

This core component of our system is responsible for data processing, model execution, and database management. Its functionalities include

1. Exposure of a TCP socket for agent connections and data reception.
2. Data sorting and processing on a nanosecond timescale to ensure precise temporal analysis.
3. Integration with the currently deployed model for real-time inference.
4. Association of model outputs with corresponding data instances.
5. Storage of processed data and detection results in a time series database for efficient querying and historical analysis.

5.3 Web-server and Dashboard

The Web-server that serves a user-friendly dashboard for system monitoring and data visualization

1. Serves a dynamic dashboard for real-time system status updates.
2. Exposes an API for querying the time series database.
3. Implements asynchronous communication between the dashboard and the API, ensuring updates every 10 seconds without impacting user experience.
4. Allows users to view and analyze historical data through interactive graphs and visualizations.

5.4 Machine Learning and Deep Learning Models

5.4.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory is a special type of recurrent neural network (RNN) designed to capture patterns in sequential data, especially in the case of long-term dependencies. This makes it perfect for time-series analysis. In our system, we use LSTM to detect DoS and DDoS attacks by learning from historical sequences of system calls and resource metrics collected from the Kubernetes cluster. The key feature of LSTM is its memory cells, which can store information for long periods. This helps it recreate the dynamic behavior of micro-services and recognize patterns that suggest an attack. By training the LSTM model on both normal and attack data, it learns to distinguish between routine operations and unusual behavior that could indicate an attack. LSTM's ability to handle evolving attack patterns makes it a strong tool for early detection of sophisticated threats in cloud native environments.

5.4.2 Gated Recurrent Unit (GRU)

Gated Recurrent Unit is another type of RNN that is great for analyzing sequential data. However, GRU has a simpler structure as compared to LSTM, making it more efficient for real-time analysis. In our system, GRU is used to monitor time-series metrics like CPU usage, memory, and network activity to detect anomalies that could indicate DoS or DDoS attacks. GRU makes use of a gating mechanism to control the flow of information, allowing it to retain important features from past data while quickly processing new information. This makes GRU especially effective when fast detection is critical. By training the GRU on sequences of normal and anomalous behavior, it can identify both short-term spikes and long-term trends that indicate malicious activity, offering a quicker alternative to LSTM for real-time detection in Kubernetes environments.

5.4.3 Random Forests

Random Forests is an ensemble learning technique that combines several decision trees to form a “forest,” making predictions based on the majority vote of these trees. This model works well for detecting DDoS/DoS attacks because it handles complex datasets involving multiple features like CPU usage, memory, network traffic, and system calls. Random Forests can effectively manage high-dimensional data by selecting random subsets of features at each decision point, ensuring it captures important patterns to differentiate between normal and malicious traffic. Additionally, Random Forests are robust against over-fitting, especially in noisy or imbalanced datasets, which is common in DDoS/DoS scenarios. This makes it a reliable choice for environments like Kubernetes clusters, where traffic can be very dynamic. Random Forests also provide insight into which features are most important for classification, helping to fine-tune the detection system. While more computationally intensive than simpler models, Random Forests scale well in distributed systems, making them suitable for large Kubernetes clusters.

5.4.4 Support Vector Machines (SVM)

Support Vector Machine is a powerful supervised learning model, particularly good at binary classification tasks, making it suitable for distinguishing between normal and malicious traffic. SVM is great for classifying data with complex boundaries, using a technique called the kernel trick to map input data into higher-dimensional spaces. This is useful in DDoS/DoS detection, where normal and attack traffic might not be easily separable. SVM is known for its accuracy, especially when the correct kernel (e.g., radial basis function or polynomial) is used. It excels in cases where there’s a clear separation in the data, as is often the case with DoS/DDoS traffic patterns. SVM is also adaptable, with various parameters (like the choice of kernel) that can be tuned to fit different network conditions and attack scenarios. While SVMs can be computationally expensive with large datasets, their precision and adaptability make them a strong candidate for DDoS/DoS detection. They offer resilience to over-fitting, which is important in noisy environments like Kubernetes clusters.

5.5 Dataset

Our dataset is crucial for training and evaluating the DoS and DDoS detection models. Key characteristics include:

1. Privately collected data from a Kubernetes cluster consisting of 4-5 nodes.
2. Utilization of an open-source micro-services application to simulate real-world scenarios.
3. Comprehensive collection of system call information and resource metrics across the entire cluster.
4. Inclusion of both normal traffic patterns and simulated attack scenarios.
5. Labeling of data to distinguish between normal cluster operations and various attack types.
6. Pending decision on data granularity (per-node vs. per-pod), to be determined based on initial experiments and performance considerations.

This dataset will provide a robust foundation for training our Machine Learning and Deep Learning models, ensuring their effectiveness in detecting DoS and DDoS attacks in Kubernetes-based micro-services environments.

6 Core Technologies

1. Golang for:
   (a) Development of Kubernetes (system calls, resource metrics) agent service.
   (b) For the development of Web-server.
   (c) Sysdig and Falco in conjunction with Golang for extracting data from the cluster

2. A Timeseries database for storing system data and any associated events.
3. Python for the development deployment of Machine Learning and Deep Learning models.
4. React.js with D3.js/Chart.js for the online dashboard.

7 Project Timeline

Elapsed Time in months since the start of the project	**Milestone**	**Deliverable**
Month 1	Project Domain + Idea Brainstorming	Project Description
Month 2	Data Collection Script Development	Dataset
Month 3	Kubernetes Agent Service Implementation	Data Collection Service
Month 4	1st Machine Learning Model Implementation	Comparative Analysis and Performance Metrics
Month 5	2nd Machine Learning Model Implementation	Comparative Analysis and Performance Metrics
Month 6	3rd Deep Learning Model Implementation	Comparative Analysis and Performance Metrics
Month 7	4th Deep Learning Model Implementation	Comparative Analysis and Performance Metrics
Month 8	Comparative Analysis Deduced and Implemented in System	Prototype and Research Paper

Table 1: Project Key Milestones & Deliverables

8 Project Equipment Details

Item(s) Name	**Type**	**No. of Units**	**Per Unit Cost (in Rs.)**	**Total (in Rs.)**
Desktop Computers with Linux	PC	5	NIL (Already on campus)	NIL
NVIDIA RTX Series GPUs	GPU	X	NIL (Already on campus)	NIL
			Total (in Rs.)	NIL

Table 2: Equipment Cost Details

References

[1] Y. Lin, O. Tunde-Onadele, and X. Gu, "Cdl: Classified distributed learning for detecting security attacks in containerized applications," *Proceedings of the 36th Annual Computer Security Applications Conference*, 2020.

[2] G. Darwesh, "Novel approach to feature collection for anomaly detection in kubernetes environment and agent for metrics collection from kubernetes nodes," *SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS*, 2023.

[3] V. Forsberg, "Automatic anomaly detection and root cause analysis for microservice clusters," 2019.

[4] Z. Zou, Y. Xie, K. Huang, G. Xu, D. Feng, and D. Long, "A docker container anomaly monitoring system based on optimized isolation forest," *IEEE Transactions on Cloud Computing*, vol. 10, no. 1, pp. 134–145, 2022. DOI: 10.1109/TCC.2019.2935724.

[5] Q. Du, T. Xie, and Y. He, "Anomaly detection and diagnosis for container-based microservices with performance monitoring," in *Algorithms and Architectures for Parallel Processing*, J. Vaidya and J. Li, Eds., Cham: Springer International Publishing, 2018, pp. 560–572, ISBN: 978-3-030-05063-4.

[6] V.-G. I. Anemogiannis, "Anomaly detection and prediction on kubernetes resources," 2023.

[7] C.-W. Tien, "Kubanomaly: Anomaly detection for the docker orchestration platform with neural network approaches," *SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS*, 2019.

[8] E. Harlicaj, "Anomaly detection of web-based attacks in microservices," 2021.