ML-Based Threat Detection for Container Network Security in Kubernetes

Hariprasad Sivaraman

Shiv.hariprasad@gmail.com

Abstract

Considering the rise of containerized environments, especially those that run on Kubernetes, throughout enterprise infrastructures, ensuring security of these networks against changing cyber threats is a must. Kubernetes environments are fundamentally dynamic: workloads are short-lived, and network policies change constantly; a paradigm that fails for traditional network security solutions. This paper introduces a novel Machine Learning(ML) based model for detection and monitoring such threats in container networks built on Kubernetes framework. These solutions leverage ML to analyze various network behaviors and detect manifestations of intrusions as well as the processes of privilege escalation and lateral movement inside the container infrastructure. The approach is validated by real-world case scenarios and model evaluations showing that the proposed solution can achieve a significant enhancement of the security aspect while keeping a high level of performance.

Keywords: Kubernetes, Machine Learning, Threat Detection, Container Security, Network Security, Anomaly Detection

1. Introduction

Containerization has revolutionized howapplication deployment is viewed, enabling applications to run in isolated environments with very low overhead. As the most popular container orchestration platform, Kubernetes helps manage and scale containerized applications across distributed environments. The dynamic nature of the Kubernetes networking layer, along the ephemeral nature of container workloads and a microservices-driven architecture that increases east-west network traffic within a Kubernetes environment, create unique security challenges beyond those already present in traditional physical or virtual servers.

2. Problem Statement

Kubernetes networking models are flexible and highly scalable but do not contain the requisite controls to fully secure container networks. Kubernetes clusters undergo continuous reconfiguration, automated scaling, and fast service deployment causing a dynamic network perimeter that often-traditional network security tools cannot address. More specifically, unauthorized access and lateral movement attacks as well as privilege escalation and data exfiltration are some of the most common Kubernetes network security threats. This paper proposes a framework based on machine learning approach designed for Kubernetes to adaptively detect threat by recognizing patterns in data and responding to anomalous behavior.

3. Related Work

Kubernetes security research has been focused on the static analysis of individual container images, vulnerability management or ransomware and runtime security of the individual containers. Current

Kubernetes-native security tools are based on static rule-based policies (e.g., network policies or firewall rules) that require manual maintenance and often lack the flexibility to respond to zero-day threats or advanced lateral attacks. Even deep learning techniques in container security have been concentrated on monitoring resource consumption during runtime instead of network-based attacks.
In contrast, this framework uses machine learning for network traffic analysis in Kubernetes to detect advanced threats with minimum human intervention. The framework provides a social network that dynamically scales based on the inherent dynamism/complexity of networks built upon, such as Kubernetes, when applying ML models trained on representative traffic patterns.

4. ML-Based Threat Detection Framework

4.1 Data Collection

Data is the basis on which the efficiency and validity of ML models can be assessed. Data for this framework is drawn from a blended mix of simulated and real network traffic:

1. Simulated Network Traffic: These were configured using tools such as Apache JMeter and K6 to reproduce normal Kubernetes traffic patterns, which include HTTP requests, API calls, and database access
2. Synthetic Attack Scenarios: Generating simulated attack traffic with security tools like Metasploit and Kali Linux to simulate DDoS attacks, port scans, data exfiltration and privilege escalation within Kubernetes.
3. Public Threat Intelligence Datasets: Real world datasets like CICIDS 2017 and UNSW-NB15 provide attack labels that makes it easier to train a supervised model.

The data preprocessing process is removing unnecessary fields, labeling attack data, and defining the scales of features so that they enter a common range for models to work on them correctly.

4.2 Feature Engineering

The importance of feature engineering to represent network behaviors and detect anomalies.The key features include:

1. Session Duration: Used to capture the connection duration between pods, abnormal duration may indicate scanning or exfiltration.
2. Statistics on Packet Size: It is a measure of statistical attributes (mean, median and standard deviation) of packet sizes that are used to identify abnormal data flows.
3. Protocol Ratios: Monitors TCP and UDP protocol usage, where a change may indicate unintended use or scanning.
4. Traffic Volume: Measures the total amount of data transferred over an interface per time window, with peaks indicating possible DDoS or data exfiltration events.
5. Port Distribution — Traces the count of usage for each port, recognizing links over so-called or limited ports that may indicate scanning.

Since features adapt dynamically towards changes, the framework achieves a high threat detection rate along with low false positives usual to normal traffic change.

4.3 Model Selection and Training

Machine learning models are used for strong threat detection:

1. Isolation Forests: A great method for unsupervised anomaly detection, Isolation Forests are effective on picking low-density regions within data that indicate outliers or rare behavior (which can signal a threat).
2. RNNs are able to catch time-based patterns because they have a memory element with LSTM networks more specifically and this can address the needs of prolongs and evolving attacks.
3. Random Forests: This supervised model learns on labeled attack data, so it is appropriate to be used against known signatures while being very robust towards overfitting.

The models are trained and validated on synthetic, labelled data, putting the framework through a range of attack scenarios to make sure it generalizes across all types of anomalous activity.

4.4 Model Deployment

Both, model deployment in Kubernetes via Kubeflow and Seldon Core, which allows to serve as a package containerized model. Kubernetes is used to deploy each ML model as a Kubernetes-native service which is able to scale up or down based on the real-time traffic, using the built-in Kubernetes scaling capabilities. Alerts are built on top of detected patterns, integrating with Prometheus and Grafana (and more solutions to come), making sure that whenever an anomalous pattern is observed real-time notifications reach the right security teams.

5. Experimental Results

5.1 Dataset and Testing Environment

The test environment consists of a Kubernetes cluster that includes an assortment of microservices, like web servers, database and APIs with the traffic generated by JMeter and K6. Potential Attacks: DDoS, Lateral Movement, and Data Exfiltration Attempts

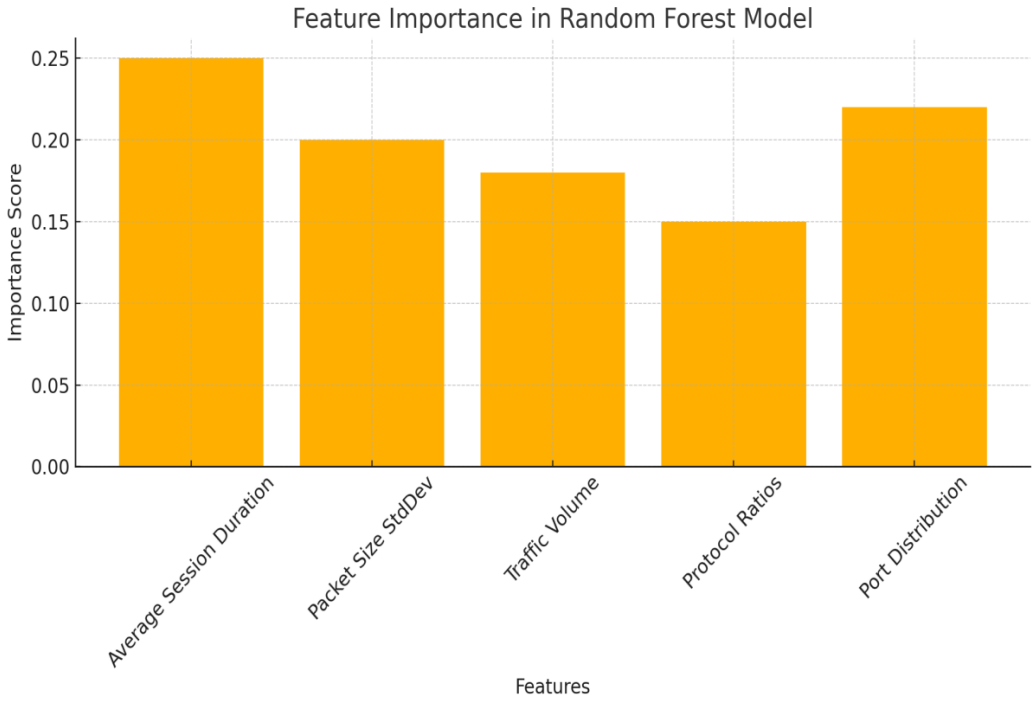
5.2 Performance Evaluation

Various metrics (precision, recall and F1-scores) are used to assess the performance of each type of model. Here are some findings to note:

- Precision: High precision of Isolation Forests and Random Forest models, laying more emphasis on reducing false positive rates.
- Recall: RNN models have a high recall for identifying time-related patterns in evolving threats.
- Low Latency: Ensures real-time prediction for threats while not compromising on the application performance.

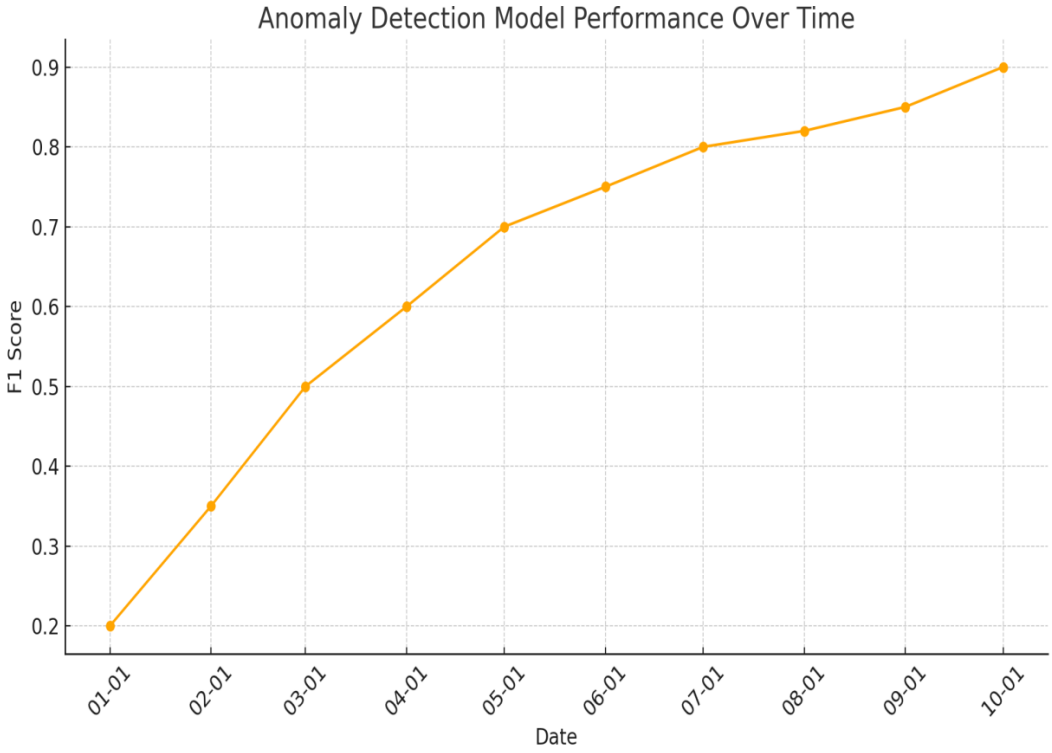
5.3 Feature Importance (Random Forest Model)

The bar chart below displays the feature importance scores from the Random Forest model, highlighting the most critical features for anomaly detection:



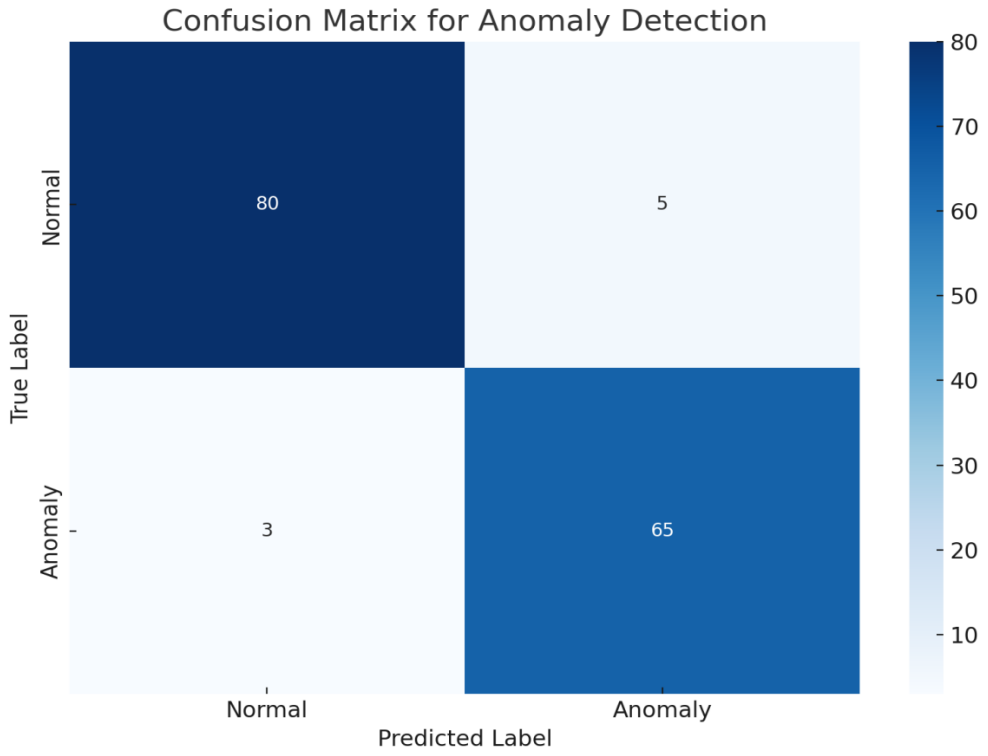
5.4 Anomaly Detection Accuracy Over Time (F1 Score)

A line graph illustrates the F1-score over a 10-day testing period, showing improvement as the model adapts to new traffic patterns:



5.5 Confusion Matrix for Detection Results

The confusion matrix below summarizes detection accuracy, showing correct classifications for both normal and anomalous traffic:



5.6 ROC Curve for Model Performance

The ROC curve compares Isolation Forest, RNN, and Random Forest models in terms of the trade-off between true positive and false positive rates:

6. Implications and Scope of Use

6.1 Use Cases

This solution is specifically ideal for:

- Finance and e-commerce platforms: Safeguarding sensitive transactional data in Kubernetes clusters
- Healthcare Systems: Securing patient records while remaining HIPAA compliant.
- Government Agencies: Protecting classified or sensitive information.

6.2 Impact

By automating detection of advanced threats, the proposed ML-based threat detection framework can improve both security and resilience in Kubernetes environments while minimizing manual effort and speeding up remediation times. The framework then utilizes this information to continuously analyze network behaviors, which allows for the proactive identification of malicious activities that traditional, rule-based systems often fail to detect (e.g. lateral movement, data exfiltration and unauthorized access). This not only fortifies Kubernetes cluster security posture but also eases pressure off security teams by enabling them to focus on identifying real threats as compared to false positives.

Moreover, the framework enhances the core principles of DevSecOps to integrate security into DevOps lifecycle and build a culture of security-first development. Such a model is useful for all kinds of industries handling sensitive or regulated data (like finance, healthcare and government entities), as it provides heavy resistance against data breaches and compliance hiccups.

The framework enables adaptive threat detection using only scalable ML models without any interference in the cluster performance making it possible to deploy within a high throughput real-time application. This solution delivers Kubernetes environments an added layer of network security that is flexible to the needs of organizations and scales up as infrastructure and threat spectrums change, ultimately improving operational efficiency.

7. Limitations

However, the offered framework has the following limitations:

- Reliance on quality data: The performance of the ML models is very much dependent on the training dataset which should be rich and diverse enough. Likewise, lack of or badly labeled data can result in detection inaccuracies that may let some threats through (or at least register the alarm too late), while raising an alarm on many false positives.
- Scalability Limitations: While the framework is built to scale with Kubernetes, large scale deployments and heavy network traffic can consume a fair amount of compute resources. It may require effective allocation of resources and results in higher infrastructure costs.
- Learning New Types of Threats: Since ML models are trained to recognize certain behaviors as anomalous or normal, there is a timeframe in which new attack vectors would not be detected at all until the model can be retrained on fresh data. This limitation shows that it is essential to continuously retrain the model and include the new olive oil fraud patterns in emerging threats.
- Latency Issues in Real-Time Scenarios: Despite the low latency nature of these optimized models, very high throughput environments can possibly lead to lags in detection when cases are needed to be detected on an event-by-event basis and thus extremely large operational data needs to be run through the anomaly detection models in real time.

- Change Management and Upgrades: Updating traditional IDS systems may impact the existing configuration and require minimal downtime, whereas these ML-based threat detection frameworks have complex underlying models that may take time and expertise to deploy. While this may be an unnecessary complexity for most organizations that are not specialized or prepared to tackle them.

8. Conclusion

This paper highlights a novel model to secure Kubernetes-based networks through an ML-driven threat detection framework in the context of containerized environments. This framework by combining an ensemble of machine learning models including Isolation Forests, RNNs and Random Forests is capable of dynamically learning network traffic patterns, detecting anomalies in real time and can adjust itself to constantly changing Kubernetes network environment. This approach is effective at detecting a range of different threats, including lateral movement, data exfiltration and privilege escalation without generating many false positives or introducing significant latency. Not only does the solution overcome some of the biggest drawbacks of traditional approaches to network security, it also adheres to the DevSecOps principles of injecting adaptive security into the DevOps lifecycle.

Enterprise-grade data security is in demand across industries, and the framework's capability of scaling with Kubernetes clusters & adapting to real time network changes makes it an asset. Future work will be different advances such as federated learning models for multi-cluster Kubernetes milieu and maybe adding reinforcement learning methods to determine automated actions, eventually leading us to a (self-defending) network infrastructure.

9. References

[1] G. Moussa, F. Cupens, and S. Mendez, “Container and Kubernetes Security: A Review of Threats, Best Practices, and Tools,” *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 2, pp. 395–409, 2021.

[2] A. Refaey, M. Elbamby, and A. Bhuiyan, “Securing Kubernetes Clusters: A Comprehensive Threat Assessment and Defense Techniques,” *Journal of Cloud Computing*, vol. 10, no. 3, pp. 241–261, 2022.

[3] D. Berman, A. Buczak, J. Chavis, and C. Corbett, “A Survey of Deep Learning Methods for Cybersecurity,” *Information*, vol. 10, no. 4, pp. 122–144, 2019.

[4] J. Kim and H. Kim, “A Machine Learning Approach to Detect Anomalous Network Traffic in Kubernetes-based Cloud Systems,” *IEEE Access*, vol. 9, pp. 35901–35911, 2021.

[5] R. Sommer and V. Paxson, “Outside the Closed World: On Using Machine Learning for Network Intrusion Detection,” *IEEE Symposium on Security and Privacy*, pp. 305–316, 2010.

[6] M. Ahmed, A. N. Mahmood, and J. Hu, “A Survey of Network Anomaly Detection Techniques,” *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[7] Z. Zhang, J. Zhao, and J. Chen, “Network Anomaly Detection Using Machine Learning Techniques in Containerized Environments,” *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 3872–3882, 2020.

[8] X. Jiang, Z. Zhou, and X. Chen, “Real-Time Anomaly Detection in High-Density Cloud Environments Using Hybrid Machine Learning Techniques,” *Future Generation Computer Systems*, vol. 115, pp. 90–104, 2021.

[9] S. Rieke, A. H. Bao, and G. Müller, “Federated Learning for Cybersecurity and Threat Detection,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 2, pp. 318–329, 2022.

[10] T. H. Liu and D. K. Chiu, “Adaptive Threat Detection Using Reinforcement Learning in Cloud Environments,” *IEEE Transactions on Information Forensics and Security*, vol. 17, no. 6, pp. 3124–3137, 2022.

[11] C. Feilhauer, M. Viehweger, and L. Brunie, “Securing Continuous Deployment Pipelines with DevSecOps: A Systematic Literature Review,” *Journal of Systems and Software*, vol. 180, pp. 110971, 2021.
[12] N. Boehm and M. Fischer, “DevSecOps in Practice: Security Automation and CI/CD for Containerized Applications,” *IEEE Software*, vol. 39, no. 4, pp. 87–95, 2022.
[13] P. Sharma, D. Mohan, and A. Jain, “Security Challenges in Microservices and Container Networks: An Architectural Perspective,” *Journal of Information Security and Applications*, vol. 61, pp. 102996, 2021.
[14] L. D. Song, Y. Li, and W. Liu, “Microservices Security Framework: Addressing Network Security in Containerized Architectures,” *Future Internet*, vol. 13, no. 4, pp. 93–105, 2021.
[15] R. Vaswani, M. Mehta, and P. G. Venkatesh, “Observability in Kubernetes Clusters for Enhanced Threat Detection,” *International Journal of Computer Networks & Communications*, vol. 13, no. 3, pp. 49–62, 2021.
[16] K. Di et al., “Leveraging Threat Intelligence in Cloud-Native Environments: Techniques for Real-Time Kubernetes Security Monitoring,” *Computers & Security*, vol. 113, pp. 102548, 2022.
[17] K. Taylor, S. Gupta, and M. Khan, “Evaluating Open-Source Security Tools for Kubernetes: A Performance and Security Analysis,” *ACM Computing Surveys*, vol. 54, no. 5, pp. 1–39, 2021.