Towards Robust Kubernetes Security: Dataset Development and Comparative Analysis of Intrusion Detection Algorithms

Exposé for Bachelor Thesis - Adrian Schubek

1 Introduction

Packaging software into containers has become a popular way to deploy and manage applications. Containers are lightweight, portable, and can run on any platform that supports them. However, managing a large number of containers efficiently can be challenging. This is where platforms like Kubernetes come in. Kubernetes has become the de facto standard for container orchestration, offering tools for auto-deployment, scaling, and management of containerized applications[5]. A recent study claims that 60% of organizations are using Kubernetes in production which is expected to grow in the future[16].

While Kubernetes provides robust tools for managing containers, it also introduces complexity and thereby potential security vulnerabilities. Therefore, securing Kubernetes clusters and detecting ongoing attacks are critical.

Each of these components is essential for the proper functioning of the Kubernetes cluster. However, they are also potential targets for cyberattacks.

Several studies have been conducted on how Kubernetes components can be attacked and how this affects the cluster [14][7][15]. For example, an attacker who gains unauthorized access to the kube-apiserver can manipulate configurations or deploy malicious workloads. Similarly, compromising etcd can result in data breaches and loss of cluster integrity. While tampering with the kube-proxy can result in availability issues. Attacks on the control plane can have severe consequences, affecting the entire cluster's management and orchestration capabilities. Therefore, it is crucial to analyze the potential attack vectors and their implications on Kubernetes clusters to enhance security.

To create a realistic simulation of a real-world Kubernetes environment, we aim to develop a comprehensive setup that captures various features, such as system calls, component logs, network traffic, and resource usage by each Kubernetes component. This simulated environment will enable the collection of a wide range of data, providing a robust foundation for our analysis.

Using the captured features, we will create a multi-class labeled dataset based on different attack types. To ensure the dataset is extensive, we will include detailed information about every single attack. In this study, we also consider different types of data representation of the dataset, taking into account the specific machine learning methods that can be utilized for Kubernetes intrusion detection. By exploring various representations, we aim to optimize the dataset's compatibility and performance with different machine learning models

To further enhance the accuracy of our labels, we will analyze and incorporate ten state-of-the-art Kubernetes attacks into the dataset. These attacks will need to be studied in depth, and including them in our dataset will enable us to create a robust and realistic set of labels. The resulting multiclass dataset will be useful for training a machine learning model in the future for identifying attacks on Kubernetes environments.

The contributions of this thesis can be summarized as follows:

- **Attack Analysis**: An in-depth study of ten state-of-the-art attacks targeting Kubernetes components, offering insights into potential vulnerabilities and their impacts.
- **Comprehensive Dataset Development**: Based on these attacks a multi-class labeled dataset will be created by capturing various features such as System calls, component logs, network traffic, and resource usage.
- **Comparative Study of Intrusion Detection Models**: A thorough analysis of the performance of various machine learning-based intrusion detection methods will be conducted, comparing the efficiency of structures such as decision trees, SVMs, random forests, and deep learning models in detecting Kubernetes threats.
- **Dataset Representation Optimization**: Multiple data representation techniques, including raw system logs, network traffic, and aggregated metrics, will be explored to optimize the compatibility and performance of machine learning models with the developed dataset.

2 Motivation

At a glance a Kubernetes cluster consists of worker machines also called nodes which run containerized applications. The control plane provide various utilities to manage the cluster and its nodes. It comprises several key components, including kube-apiserver, etcd, kube-scheduler, kube-controller-manager, and cloud-controller-manager.

The kube-apiserver is a critical component that exposes the Kubernetes API, serving as the front end for the Kubernetes control plane. Another essential component is etcd, a consistent and highly-available key-value store that acts as Kubernetes storage for distributed cluster data.

The kube-scheduler plays a vital role in managing Pod placement, watching for newly created Pods with no assigned node and selecting a suitable node for them to run on. The kube-controller-manager is responsible for running controller processes, which monitors the state of the clustzer and makes changes to bring the actual state closer to the desired state.

Finally, the cloud-controller-manager enables the linking of the cluster to the cloud provider's API, separating the components that interact with the cloud platform from those that interact with the cluster.

On each worker node, the kubelet component is responsible for managing the pods and containers running on that node. The kube-proxy is used for maintaining network rules on the host and performing connection forwarding. And finally the container runtime, which is responsible for running the actual containers.

Kubernetes clusters are prime targets for a variety of cyberattacks. Understanding the potential attack vectors and their implications on the cluster is important for enhancing security. In this context, we explore the types of attacks that can be executed against the cluster, the specific core components that are vulnerable, and the impact on the cluster's performance and availability.

The cluster can be vulnerable to several types of attacks[10], each targeting different core components. The control plane, which includes the API server, etcd, controller manager, and scheduler, is a critical area of concern. Attacks on the control plane can lead to severe disruptions, affecting the entire cluster's management and orchestration capabilities. For instance, an attacker gaining unauthorized access to the API server can manipulate configurations, deploy malicious workloads, or even shut down the cluster. Similarly, compromising etcd, the key-value store that holds the cluster's state, can result in data breaches and loss of cluster integrity.

On the other hand, attacks on individual worker nodes, which host the actual application workloads (pods), can also be detrimental but in a different manner. Compromising a worker node can lead to resource exhaustion, unauthorized access to sensitive data within pods, and potential lateral movement within the cluster[13].

The broader impact of these attacks on the cluster's overall security and stability, beyond individual pods, remains underexplored and requires further investigation. While the immediate consequences may be localized, the performance and availability of services across the cluster can still be significantly degraded. This study intends to shed light on these vulnerabilities, aiming to enhance our understanding and develop more robust mechanisms against threats in Kubernetes environments.

Initial access	Execution	Persistence	Privilege escalation	Defense evasion	Credential access	Discovery	Lateral movement	Collection	Impact
Using Cloud	Exec into Container	Backdoor Container	Privileged Container	Clear Container Logs	List K8s secrets	Access the K8s API server	Access cloud resources	Images from a private repository	Data Destruction
Compromised images in registry	bash/cmd in container	Writable hostPath mount	Cluster-admin binding	Delete k8s events	Mount service principal	Access Kubelet API	Container service account		Resource hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod / container name similarity	Access container service account	Network mapping	Cluster internal networking		Denial of Service
Application vulnerability	Application exploit (RCE)	Malicious admission controller	Access cloud resources	Connect from proxy server	Applications credentials in configuration files	Access Kubernetes dashboard	Applications credentials in configuration files		
Exposed sensitive interfaces	SSH server running in inside container		Disable Namespacing		Access managed identity credentials	Instance metadata API	Writable volume mounts on the host		
	Sidecar injection				Malicious admission controller		CoreDNS poisoning		
							ARP poisoning and IP spoofing		

Figure 1. Kubernetes Threat Matrix

3 Related Work

The threat matrix in Figure 1 provides an overview of the different attack vectors and vulnerabilities in Kubernetes clusters[3]. This matrix by Microsoft is based on the popular MITRE ATT&CK framework knowledge base which provides a comprehensive list of potential threats and their corresponding techniques[4].

There are several papers that discuss attacks on Kubernetes clusters. For example, Ronen Ben David et al. [11] discuss a specific attack on the Kubernetes autoscaler. The paper shows that the Kubernetes autoscaler can be exploited to launch a denial of service attack on the cluster. Likewise Sever et al. [2] researched different CVEs in popular open-source software, how they can be exploited and created a small dataset. However they focused on individual application software and not the cluster components itself.

Noah Spahn et al. [15] created a honeypot to detect attacks on Kubernetes clusters. The honeypot was able to detect attacks on components of the control plane and worker nodes such as the api server, kubelet and container runtime. They found that through specially crafted requests to the api-server endpoints, attackers could sucessfully execute a privilege escalation attack leading to persistent access to the cluster[6].

A security audit by Robert Tonic et al.[1] of the etcd key-value store used in Kubernetes revealed several different vulnerabilites. Some leading to unauthorized access to the etcd key-value store and others to potential denial of service attacks. A compromised node hosting the etcd service can, under certain cirmumstances, even lead to a complete cluster takeover[7].

Similarly, according to paper by Carmen Carrion et al., the Kubernetes scheduler is limited in its ability to consider security aspects during the scheduling process. The default Kubernetes scheduler primarily focuses on allocating resources such as CPU and memory, without taking into account the security implications of deploying containers on specific nodes. This lack of security awareness can lead to potential vulnerabilities and threats to the system, as containers may be scheduled on nodes that exhibit suspicious behavior e.g. they are under attack[9].

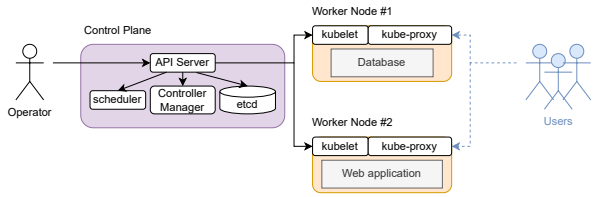

Figure 2. Kubernetes Cluster Architecture

4 Methodology

In the study of security vulnerabilities within Kubernetes clusters, an experimental cluster setup will be created consisting of a control plane and two worker nodes. The control plane incorporates essential Kubernetes components such as the kube-apiserver, kube-controller-manager, kube-scheduler, and etcd. The worker nodes will host different production applications to simulate a realworld deployment.

The focus of the research centers on the execution and analysis of ten state-of-the-art attacks aimed at different components of the Kubernetes cluster. These attacks are carefully chosen to provide a broad and in-depth exploration of potential security weaknesses across both the control plane and the worker nodes. By systematically targeting specific components such as the kube-apiserver, valuable insights can be gained into the resilience of the cluster and the potential points of failure.

The Goal is not to find or exploit vulnerabilities in indiviudal containers like nginx or mongodb but to attack the cluster core components like the kube-apiserver itself. Attacks on these components can have a much larger impact on the cluster and its worker nodes than attacks on individual pods as they affect multiple worker nodes and the cluster as a whole.

To effectively measure the impact of these attacks, a comprehensive set of metrics is employed. System calls, component logs, network traffic, and resource usage are tracked to measure the state of the cluster. Additionally, security-specific metrics such as unauthorized access attempts, configuration changes, and anomalies in network traffic are recorded. These metrics not only quantify the direct effects of the attacks but also contribute to an understanding of how Kubernetes clusters respond to various threats.

An integral part of this research involves the extensive collection and preparation of data to create a robust dataset that encapsulates both normal operational parameters and the effects of malicious activities. Traffic data is captured using packet sniffing tools, followed by preprocessing of the PCAP data to extract relevant features. Likewise, resource usage will be monitored using utilities like Prometheus. Subsequently, this data is organized and stored in a structured format within a database, facilitating efficient analysis and retrieval. This dataset includes timestamps, resource usage metrics, network traffic patterns, and detailed logs, providing a rich foundation for further analysis.

The compiled data serves as a training ground for developing machine learning models capable of detecting intrusion threats against a Kubernetes environment. By leveraging this dataset, models are trained to recognize a range of attack signatures and their impacts, enhancing the predictive capabilities of security measures in place. The explainability aspect of the model is also considered by using a wide range of features, ensuring that the models can provide insights into the underlying reasons for their predictions. This research aims to contribute to the ongoing efforts to secure Kubernetes clusters and fortify their resilience against potential cyber threats.

In addition to developing a comprehensive multi-class dataset for Kubernetes intrusion detection, this thesis will also benchmark state-of-the-art machine learning methods for detecting such threats. By evaluating the dataset against several widely-used intrusion detection models, such as decision trees, support vector machines, random forests, and deep learning approaches like neural networks[12][8], we will assess the efficacy of these models in identifying different types of attacks. This not only ensures that the dataset is robust but also contributes to related work by comparing and contrasting various machine learning techniques in the context of Kubernetes security. By integrating this comparative analysis, the thesis aims to provide a deeper understanding of the strengths and limitations of current machine learning approaches in securing Kubernetes environments, thus enriching the field with practical insights.

While we will investigate various data representations, the development of a machine learning model for intrusion detection lies beyond the scope of this thesis.

References

[1] Etcd/security/SECURITY_AUDIT.pdf at main · etcd-io/etcd. https://github.com/etcd-io/etcd/blob/main/security/SECURITY_AUDIT.pdf.

[2] A Kubernetes dataset for misuse detection. https://www.itu.int:443/en/publications/gs/Pages/publications.aspx, .

[3] Kubernetes Threat Matrix. https://kubernetes-threat-matrix.redguard.ch/, .

[4] MITRE ATT&CK®. https://attack.mitre.org/.

[5] Production-Grade Container Orchestration. https://kubernetes.io/.

[6] Kubernetes RBAC: How to Avoid Privilege Escalation. https://www.aquasec.com/blog/kubernetes-rbac-privilige-escalation/, April 2022.

[7] Post-exploiting a compromised etcd – Full control over the cluster and its nodes. https://research.nccgroup.com/2023/11/07/post-exploiting-a-compromised-etcd-full-control-over-the-cluster-and-its-nodes/, November 2023.

[8] Zeeshan Ahmad, Adnan Shahid Khan, Cheah Shiang, and Farhan Ahmad. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32, January 2021. doi: 10.1002/ett.4150.

[9] Carmen Carrión. Kubernetes Scheduling: Taxonomy, Ongoing Issues and Challenges. *ACM Comput. Surv.*, 55(7):138:1–138:37, December 2022. ISSN 0360-0300. doi: 10.1145/3539606.

[10] Ghadeer Darwesh, Jaafar Hammoud, and A.A. Vorobeva. SECURITY IN KUBERNETES: BEST PRACTICES AND SECURITY ANALYSIS. 2: 63–69, June 2022. doi: 10.14529/secur220209.

[11] Ronen Ben David and Anat Bremler Barr. Kubernetes Autoscaling: YoYo Attack Vulnerability and Mitigation. In *Proceedings of the 11th International Conference on Cloud Computing and Services Science*, pages 34–44, 2021. doi: 10.5220/0010397900340044.

[12] Satish Kumar, Sunanda Gupta, and Sakshi Arora. Research Trends in Network-Based Intrusion Detection Systems: A Review. *IEEE Access*, 9:157761–157779, 2021. ISSN 2169-3536. doi: 10.1109/ACCESS.2021.3129775.

[13] Francesco Minna, Agathe Blaise, Filippo Rebecchi, Balakrishnan Chandrasekaran, and Fabio Massacci. Understanding the Security Implications of Kubernetes Networking. *IEEE Security & Privacy*, 19(5):46–56, September 2021. ISSN 1558-4046. doi: 10.1109/MSEC.2021.3094726.

[14] Md Shazibul Islam Shamim, Farzana Ahamed Bhuiyan, and Akond Rahman. XI Commandments of Kubernetes Security: A Systematization of Knowledge Related to Kubernetes Security Practices, June 2020.

[15] Noah Spahn, Nils Hanke, Thorsten Holz, Christopher Kruegel, and Giovanni Vigna. Container Orchestration Honeypot: Observing Attacks in the Wild. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 381–396, Hong Kong China, October 2023. ACM. ISBN 9798400707650. doi: 10.1145/3607199.3607205.

[16] Statista Statista. Topic: Kubernetes. https://www.statista.com/topics/8409/kubernetes/.