Dynamic Application Security Testing for Kubernetes Deployment: An Experience Report from Industry

Shazibul Islam Shamim
mshamim@kennesaw.edu
Kennesaw State University
Marietta, Georgia, USA

Hanyang Hu
phenom.hu@gmail.com
Company-Z
San Francisco, California, USA

Akond Rahman
akond@auburn.edu
Auburn University
Auburn, Alabama, USA

ABSTRACT

While Kubernetes enables practitioners to rapidly deploy their software and perform container orchestration efficiently, security of the Kubernetes-based deployment infrastructure is a concern for industry practitioners. A systematic understanding of how dynamic analysis can be used for securing Kubernetes deployments can aid practitioners in securing their Kubernetes deployments. We present an experience report, where we describe empirical findings from three dynamic application security testing (DAST) tools on a Kubernetes deployment used by 'Company-Z'. From our empirical study, we find (i) 3,442 recommended security configurations are violated in 'Company-Z's' Kubernetes deployment; and (ii) of the three studied DAST tools, Kubescape and Kubebench provide the highest support with respect to detecting 14 types of recommended security configurations. Based on our findings, we recommend practitioners to apply DAST tools for their Kubernetes deployments, and security researchers to investigate how to detect configuration violations dynamically in the Kubernetes deployment.

CCS CONCEPTS

• **Software and its engineering → Software configuration management and version control systems**; *Formal software verification*; *Empirical software validation*; • **Security and privacy** → *Software security engineering*.

KEYWORDS

devops, devsecops, kubernetes, security

ACM Reference Format:

Shazibul Islam Shamim, Hanyang Hu, and Akond Rahman. 2025. Dynamic Application Security Testing for Kubernetes Deployment: An Experience Report from Industry. In *33rd ACM International Conference on the Foundations of Software Engineering (FSE '25), June 23–28, 2025, Trondheim, Norway.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3696630.3728573

1 INTRODUCTION

With the advent of container-based technologies, such as Docker, usage of containers is becoming prevalent for software deployment. In order to manage containers, practitioners apply the practice of container orchestration with tools, such as Kubernetes. Kubernetes has emerged as an open-source container orchestration that helps practitioners manage provision containers efficiently [5, 13]. Organizations that adopted Kubernetes report significant benefits of using Kubernetes. For example, practitioners at Adidas reported that using Kubernetes, deployment frequency was increased from every 4-6 weeks to 3-4 times a day. [6]. As another example, in the case of Denso, the development cycle was reduced from 2-3 years to 2 months [7].

While Kubernetes has aided in achieving multiple benefits, the security of Kubernetes deployments is a concern amongst industry practitioners. According to the '2024 State of Kubernetes Security Report' that surveyed 600 practitioners, 60% respondents reported security vulnerabilities in their Kubernetes deployments to be a concern [23]. This concern is well-founded as security vulnerabilities in Kubernetes deployments have been leveraged to conduct security attacks, such as the infamous 'Tesla attack', where Tesla's Kubernetes deployment was attacked to conduct a cryptomining attack. Cryptojacking is the attack of using a computing resource to stealthily mine cryptocurrency without the user's awareness [30].

The above-mentioned evidence showcases the importance of securing Kubernetes deployments. The practitioner community has responded to this need by deriving guidelines and developing dynamic application security testing (DAST) tools. For example, the 'Center for Internet Security (CIS)' organization has listed a set of recommended security configurations that practitioner show apply in their Kubernetes deployments [10]. CIS is a non-profit organization that recommends a set of security guidelines for various software systems based on a consensus of cybersecurity experts [9]. Multiple DAST tools, such as Trivy [27] and Kubescape [17] are available for practitioners to use.

Despite the availability of DAST tools, there is a lack of understanding on the capabilities of these tools with respect to detecting the violations of recommended security configurations. Such an understanding is pivotal as Kubernetes-related configurations similar to that of Figure 1 can occur, which violates the recommended guideline of 'Ensure that all Namespaces have Network Policies defined'. 'Company-Z' uses Kubernetes to deploy software applications, and consider the security of software deployments to be of paramount

```
kind: Pod
metadata:
   name: simple-pod
spec:
   volumes:
   - name: simple-no-ctx-vol
      emptyDir: {}
```

Figure 1: Example of a specification that violates a CIS-recommended guideline. Here, a pod does not use `securityContext`, which makes a Kubernetes deployment susceptible to security attacks [22]. A pod is the most fundamental deployment unit in Kubernetes that groups multiple containers together [16]. Use of `securityContext` is critical to restrict malicious activities that can arise from supply chain attacks for Kubernetes deployments [20].

importance. The practitioners at Company-Z have been emphasizing in applying DAST tools that detect violations of recommended security configurations. This emphasis motivates us to investigate existing DAST tools for Kubernetes deployments. Such an investigation can aid practitioners with recommendations on who to secure their Kubernetes deployments, and by providing them with evidence of commonly violated recommended configurations.

We answer the following research questions:

- **RQ1 [Support]**: *How frequently do dynamic application security testing tools support recommended security configurations in Kubernetes deployments?*
- **RQ2 [Frequency]**: *How frequently are recommended security configurations violated in a Kubernetes deployment?*

We conduct an empirical study by applying 3 DAST tools in Company-Z's Kubernetes deployment. *First*, we compute how frequently each of the 3 DAST tools support 33 CIS-recommended configurations. *Second*, we use the output of DAST tools to quantify how frequently CIS-recommended configurations are violated.

Contributions: We list our contributions as follows:

- An evaluation of support for three DAST tools with respect to detecting recommended security configurations; and
- An evaluation of how frequently recommended security configurations are violated in an organization's Kubernetes deployment.

2 RQ1: SUPPORT

First, we provide background information on Kubernetes. Then, we provide context of why 'Company-Z' adopted Kubernetes in Section 2.2. Next, we provide the methodology and results respectively, in Sections 2.3 and 2.4.

2.1 Background

Kubernetes is an open-source software for the automated management of containerized applications [21]. A Kubernetes installation is also called a Kubernetes cluster [21]. Each Kubernetes cluster contains a set of worker machines defined as nodes, and there are two types of nodes in Kubernetes: control-plane nodes and worker nodes.

Each control-plane node contains the components: 'API server', 'scheduler', 'controller', and 'etcd' [21]. Kubernetes serves its functionality through an application program interface from the 'API server'. The 'API server' is responsible for orchestrating all the operations within the cluster. Practitioners use a command-line tool, 'Kubectl,' to communicate with the 'API server' in the control plane node. The worker nodes host the applications that run on Kubernetes [21]. The following components are included in the worker node: 'kube-proxy,' 'kubelet', and 'pod.' The pod is the smallest Kubernetes entity, which includes at least one active container. A container is a standard software unit that packages the code and related dependencies to run in any computing environment [21].

2.2 Kubernetes Adoption in 'Company-Z'

'Company-Z' started using Kubernetes in 2018 to build edge computing solutions to provide a scalable, reliable, low-latency cloud platform optimized for distributed edge applications. 'Company-Z adopted Kubernetes for its core orchestration layer, which has robust container orchestration capabilities, strong ecosystem support, and the ability to meet the dynamic scaling and resource management needs of edge environments. The widespread adoption of Kubernetes and its flexibility made it a natural choice to support the goals of 'Company-Z' in standardization and interoperability in edge computing infrastructure. Using Kubernetes, the 'Company-Z' supports a deployment-ready, manageable, scalable, and highly reliable complete virtualized edge infrastructure platform for container workloads. Practitioners at Company-Z operate their entire infrastructure using Amazon Elastic Kubernetes Service (EKS), a managed Kubernetes service in the Amazon Web Services (AWS) cloud deployment.

Company-Z considers security to be an a pivotal aspect in their software development and deployment process. As such, practitioners who are involved with Kubernetes deployments, have been exploring tools that can aid in securing their Kubernetes deployments. Our experience report provides a discussion on the capabilities of DASTs with respect to coverage.

2.3 Methodology for RQ1

Our empirical study focuses on security configurations recommended by a community of cybersecurity experts who specialize in Kubernetes. Practitioners at 'Company-Z' follow the 'CIS Amazon EKS benchmark v1.2.0' guideline, which prescribes 33 configurations for Kubernetes deployments that use the Amazon EKS service for container orchestration [10]. Cybersecurity experts in the Center for Internet Security (CIS) provide these recommendations based on consensus [8]. Each recommendation is mapped to a category. For example, the recommendation 'Minimize the admission of root containers' is mapped to a category called 'Pod Security Policies'.

We performed a qualitative analysis technique called closed coding [24] to create a one-to-one mapping relation between rules or policies of the dynamic analysis tools used by the practitioners at 'Company-Z' and CIS-recommended security recommendations.

To answer RQ1, we use the three dynamic analysis tools used by 'Company-Z'. The Kubernetes practitioners at 'Company-Z' selected three tools from a set that can detect violations of CIS-recommended security recommendations in the Amazon EKS deployment. To select these tools, the practitioners considered a set of dynamic analysis tools, including two commercial tools that require paid subscription and license based on their experience and security requirements of 'Company-Z.' The practitioners applied the following criteria to identify the three tools.

- *Criterion-1*: The dynamic analysis tool must be publicly available online and free to use without any subscription or license.
- *Criterion-2*: The tool must be executable using the command line interface in the Amazon EKS deployment of 'Company-Z'. The practitioners exclude tools like Datree and Snyk as those tools require manual integration for importing and scanning the Amazon EKS deployment for security-related analysis.
- *Criterion-3*: The dynamic analysis tool must be able to detect violations of recommended configurations in the Amazon EKS. The practitioners documented that each tool can detect violations of security-related configuration related to the Amazon EKS deployment of 'Company-Z'.
- *Criterion-4*: The practitioners select the tool that detects violations of at least five security configurations. This criterion the practitioners selected is consistent with prior research on security tool evaluation that uses a minimum threshold of five security weakness types to determine the generalizability of a tool [19].

Upon application of the following criteria the practitioners identify three tools Kube-bench, KubeScape, and Trivy to analyze Amazon EKS deployment of 'Company-Z'. Attributes of these three tools are available in Table 1.

Kube-bench [14] is an OSS tool developed by Aqua Security. Kubescape can verify the Kubernetes deployment configuration with the CIS benchmark for Kubernetes. Practitioners can run Kube-bench from the command line interface as a dynamic analysis tool or as a job to scan Kubernetes deployment configurations.

Kubescape [17] is an OSS security analysis tool developed by ARMO. Kubescape provides support for misconfiguration scanning and security compliance within Kubernetes deployments. Practitioners can use Kubescape as a static analysis tool to scan source code in a local directory using a command line interface and as a dynamic analysis tool to analyze Kubernetes deployment configurations.

Trivy is an OSS tool developed by Aqua Security that can detect security weaknesses in Kubernetes configuration files, and Kubernetes deployment [27]. Practitioners can use Trivy to detect security weaknesses for known vulnerabilities, misconfigurations, and runtime security issues.

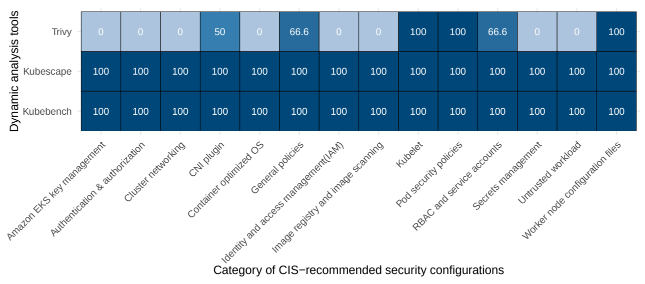

Figure 2: Answer to RQ1: Support of DAST for CIS-recommended security configurations.

2.3.1 *Map Recommendations to Rules Implemented in the Tools.*

The focus of is to identify which of the CIS-recommended security configurations are supported by which tool. We use a qualitative technique called closed coding [24] to perform a mapping between the CIS-recommended recommendations and a rule implemented within the tool. As part of applying the closed coding technique, the first author and the second author of the paper read each of the recommendations and if the recommendation security configuration is detected by inspecting the source code or documentation of the tool. The second author of the paper is practitioner at Company-Z. As part of applying closed coding we excluded recommendations that are not applicable for dynamic analysis.

2.4 Answer to RQ1

In Figure 2, we provide an overview of the support for the dynamic analysis tools used in 'Company-Z'. The CIS-recommended security configurations are divided into 14 categories. A complete breakdown of the security configurations is available in Table 3. For example, the security configuration 'Ensure that the cluster-admin role is only used where required' is used for the 'RBAC and service accounts' category. We observe KubeScape and Kubebench to have the support of 100% for 14 out of the 14 CIS-recommended security configuration categories. In the case of Trivy, we observe >= 50% support for 6 of the 14 categories. The remaining eight categories that Trivy does not cover are: 'container optimized OS', 'CNI plugin', 'Secrets management', 'pod security policies', 'Image registry and image scanning', 'Identity and access management(IAM)', 'Amazon EKS key management', 'Cluster networking', 'Authentication & authorization', and 'Untrusted workload'.

Answer to RQ1: Among the three DAST tools, Kubescape and Kubebench provide the most support for detecting violations of CIS-recommended security configurations for Kubernetes deployments.

3 RQ2: FREQUENCY

We provide the methodology and results respectively, in Sections 3.1 and 3.2.

Table 1: Attributes of Selected DAST Tools

Tool	Size (KLOC)	Source	Detection Method	Output Format
Kube-bench	63.04	GitHub [14]	Kube-bench identifies violations of CIS recommendations in Kubernetes deployment. For each CIS recommendation, Kube-bench executes specific commands in the Kubernetes deployment and reports the result	JSON, TXT
KubeScape	257.61	GitHub [17]	Kubescape defines rules using the 'rego' policy language and communicates with the Kubernetes API server using 'k8s-interface' wrapper to collect deployment resources. After matching the defined rules with the deployment resources, Kubescape identifies misconfigurations.	SARIF, JSON, XML, HTML, PDF
Trivy	514.05	GitHub [27]	Trivy also uses 'rego' policy language and communicates with API server with 'trivy-kubernetes' wrapper for collecting deployment resources. Trivy identifies vulnerabilities and misconfigurations inside the Kubernetes deployments.	SARIF, JSON, XML, HTML

3.1 Methodology for RQ2

We use the three DAST tools to identify configurations in Company-Z's Kubernetes deployment that violate CIS-recommended guidelines. Attributes of the deployment is available in Table 2. Each of the three tools is executed from the command line. Output of the tools are available as JSON files. We report the total count of violation for each of the CIS-recommended security configurations for each of the DAST tools.

Table 2: Attributes of the Amazon EKS Deployment

Attribute	Value
Number of Worker Nodes	5
Number of Namespaces	53
Total Number of Resources deployed in the deployments	2,421
Maximum number of Resources in a Namespace	664
Minimum number of Resources in a Namespace	2
Cloud Provider	AWS EKS
Region of AWS	us-east-1, us-east-2
API server version	v1.24.15-eks-a5565a6
Platform	'linux/amd64'
Environment	Production

3.2 Answer to RQ2

We report our findings in Table 3 using columns 'Count (Kubescape)', 'Count (Trivy)' and 'Count (Kube-bench)'. Each column shows the count of violated of CIS-recommended security configurations. If the tool can not detect violation then we marked the count as 'NA'. We observe the most frequently violated configuration to be 'Apply Security Context to Your Pods and Containers'. In all, we observe a total of 1482, 1955 and 5 instances of violations respectively, detected by Kubescape, Trivy and Kube-bench.

Answer to RQ2: We identify 3,442 violations of CIS-recommended configurations in the Kubernetes deployment of 'Company-Z'.

4 DISCUSSION

We discuss the implications of our findings and threats to validity respectively, in Sections 4.1 and 4.2.

4.1 Implications

4.1.1 Implications for Practitioners on Combining Dynamic Analysis Results. From Table 3, we observed that no DAST tool has comprehensive coverage for the CIS-recommended security configurations. Despite these shortcomings of individual DAST tools, overall, they detect many violations of CIS-recommended configurations. Furthermore, specific DAST tools can help detect violations of certain categories of CIS-recommended security configurations. For instance, Kubescape detects 6 CIS-recommended security configurations in the 'RBAC and Service Accounts category, while Trivy identifies 7 in the 'Pod Security Policies category. Combining the results from the DAST tools can help practitioners identify a broader range of violations related to CIS-recommended security configurations. Therefore, Kubernetes practitioners should utilize DAST tools in their Kubernetes deployments and integrate the results from these tools to enhance their deployments' security.

4.1.2 Implication for Security Researchers. From Figure 2, and Table 3, we observed that DAST tools can not detect the violation configurations even though the tools have rules to detect the violations. For instance, Kubebench supports 33 CIS-recommended configurations. However, Kubebench can only detect five CIS-recommended security configurations in the Kubernetes deployment of 'Company-Z'. Furthermore, we observed only three CIS-recommended security configurations where two tools agreed on violation detection. One potential reason for this inconsistency and the inability to detect CIS-recommended security recommendations is that the dynamic analysis tools have different rules for detecting security configuration violations. For instance, Trivy and Kubescape use different Rego-based rules [28], [18] for detecting CIS-recommended security configurations. At the same time, Kubebench runs commands with privilege inside the Kubernetes deployments [15]. As a result, the output of the DAST tools varies. Moreover, according to the CIS recommendation guideline, most configurations require manual analysis; as a result, they can not be detected automatically. Hence, we advocate that the researchers investigate how to detect configuration violations dynamically in the Kubernetes deployment.

4.2 Threats to Validity

We describe the limitations of our paper as follows:

External Validity: Our dynamic analysis tool evaluation results are based on Company-Z's Kubernetes deployment and our findings may not generalize to other Kubernetes deployment.

Conclusion Validity: The mapping between the rules of security analysis tools and recommended configurations are susceptible to rater bias. We mitigate this limitation by using two raters, of which

Table 3: Frequency of Violation for CIS-recommended Security Configurations in 'Company-Z'

Category	Recommended Configuration	Count (Kubescape)	Count (Trivy)	Count (Kube-bench)
	Ensure that the 'Anonymous Auth' is not enabled	NA	NA	1
Worker node configuration files	Ensure that the –authorization-mode argument is not set to 'AlwaysAllow'	NA	NA	1
	Ensure that the –streaming-connection-idle-timeout argument is not set to 0	NA	NA	1
Kubelet	Ensure that the –protect-kernel-defaults argument is set to true	NA	NA	1
	Ensure that the –make-iptables-util-chains argument is set to true	5	NA	1
Container optimized OS	Prefer using a container-optimized OS when possible	5	NA	NA
	Ensure that the cluster-admin role is only used where required	1	NA	NA
	Minimize the access to secrets	99	25	NA
	Minimize the wildcard use in roles and clusterroles	2	2	NA
RBAC and service accounts	Minimize access to create pods	54	NA	NA
	Ensure that default service accounts are not actively used	61	NA	NA
	Ensure that the service account tokens are only mounted where necessary	397	NA	NA
	Limit use of the bind, impersonate and escalate permissions in the Kubernetes cluster	40	NA	NA
	Minimize the admission of privileged containers	NA	6	NA
	Minimize the admission of containers wishing to share host process ID namespace	NA	2	NA
	Minimize the admission of containers wishing to share host network namespace	NA	6	NA
Pod security policies	Minimize the admission of containers with allowPrivilegeEscalation	NA	173	NA
	Minimize the admission of root containers	NA	233	NA
	Minimize the admission of containers with added capabilities	NA	4	NA
	Minimize the admission of containers with capabilities assigned	NA	224	NA
CNI plugin	Ensure that all namespaces have network policies	47	NA	NA
	Prefer using secrets as files over secrets as environment variables	47	NA	NA
Secrets management	Consider external secret storage	256	NA	NA
	Create administrative boundaries between reources using namespaces	49	NA	NA
General policies	Apply security context to Your Pods and Containers	263	1,277	NA
	The default namespace should not be used	21	NA	NA
	Ensure Image Vulnerability Scanning using Amazon ECR image scanning or a third party provider	1	NA	NA
Image registry and image scanning	Minimize user access to Amazon ECR	1	NA	NA
Identity and access management (IAM)	Prefer using dedicated EKS Service Accounts	51	NA	NA
AWS EKS key management service	Ensure Kubernetes are encrypted using custom master keys (CMKs) managed in AWS KMS	1	NA	NA
Cluster networking	Encrypt traffic to HTTPS load balancers with TLS certificates	2	NA	NA
Authentication & authorization	Manage Kubernetes RBAC users with AWS IAM Authenticator for Kubernetes	74	NA	NA
Untrusted Workload	Consider Fargate for running untrusted workloads	5	NA	NA
All	-	**1,482**	**1,955**	**5**

one is an industry practitioner with experience in cybersecurity. Construction of our evaluation dataset is also susceptible to rater bias. We mitigate this limitation by using two raters.

Internal Validity: One of the raters who performs the mapping is a practitioner working at Company-Z. This affiliation may affect intuitively affect the mapping process. We mitigate this limitation by using another rater who has no affiliation with Company-Z.

5 RELATED WORK

Our paper is related with prior research that have addressed Kubernetes security and security tool evaluation.

To enhance the security of Kubernetes deployments, researchers have implemented anomaly-based approaches [1, 26]. Tien et al developed an anomaly detection tool that can monitor and detect attacks in the Kubernetes deployment [26]. Cao et al. proposed an anomaly detection tool that uses a state machine model for Kubernetes deployment [4]. Hariri et al. developed an anomaly detection tool specifically for scientific applications runnning in Kubernetes [12].

Additionally, researchers used graph-based approaches to secure Kubernetes deployments. Blaise et al. proposed a graph-based approach to extract and identify the attack path for Kubernetes deployment [2]. Haque et al. constructed a knowledge graph for automating security configurations and mitigating misconfigurations in Kubernetes deployment [11]. Zhu et al. developed an automated security policy tool that can protect applications in the Kubernetes deployment at runtime [29]. Empirical insights are another perspective for Kubernetes security research. Bose et al. conducted qualitative analysis and constructed a dataset with security-related commits [3]. Rahman et al. [22] developed a security analysis tool called SLI-KUBE for detecting misconfiguration [22]. A followup evaluation of the tool showed to be inadequate for detection of CIS-recommended configurations [25].

However, the discussion above showcases a lack of research in the DAST tool evaluation for the Kubernetes deployment. We address this research gap in our paper.

6 CONCLUSION

Despite CIS-recommended security configurations for securing Kubernetes deployments, there is a lack of understanding of the extent to which DAST tools detect violations of these recommendations. We have conducted an empirical study with 33 CIS-recommended configurations and identified 3,442 configuration violations in the Amazon EKS deployment of 'Company-Z'. Based on our findings, we recommend practitioners utilize DAST tools to secure Kubernetes deployments.

ACKNOWLEDGMENTS

We thank the PASER group at Auburn University for their valuable feedback. This research was partially funded by the U.S. National Science Foundation (NSF) Award # 2247141 and Award # 2312321. This work has benefitted from Dagstuhl Seminar 23082 "Resilient Software Configuration and Infrastructure Code Analysis."

REFERENCES

[1] Josue Genaro Almaraz-Rivera. 2023. An Anomaly-based Detection System for Monitoring Kubernetes Infrastructures. *IEEE Latin America Transactions* 21, 3

(2023), 457–465. https://doi.org/10.1109/TLA.2023.10068850

[2] Agathe Blaise and Filippo Rebecchi. 2022. Stay at the Helm: secure Kubernetes deployments via graph generation and attack reconstruction. In *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*. IEEE, 59–69.

[3] Dibyendu Brinto Bose, Akond Rahman, and Shazibul Islam Shamim. 2021. 'Under-reported' Security Defects in Kubernetes Manifests. In *2021 IEEE/ACM 2nd International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)*. IEEE, 9–12.

[4] Clinton Cao, Agathe Blaise, Sicco Verwer, and Filippo Rebecchi. 2022. Learning State Machines to Monitor and Detect Anomalies on a Kubernetes Cluster. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*. 1–9.

[5] Carmen Carrión. 2022. Kubernetes Scheduling: Taxonomy, Ongoing Issues and Challenges. *ACM Comput. Surv.* 55, 7, Article 138 (dec 2022), 37 pages. https://doi.org/10.1145/3539606

[6] Case Study: Adidas. 2025. Kubernetes. https://kubernetes.io/case-studies/adidas/. [Online; accessed 13-January-2025].

[7] Case Study: Denso. 2025. Kubernetes. https://kubernetes.io/case-studies/denso/. [Online; accessed 13-January-2025].

[8] Center for Internet Security(CIS). 2023. CIS Amazon EKS Kubernetes Benchmark v1.2.0. https://www.cisecurity.org/benchmark/kubernetes. [Online; accessed 22-november-2023].

[9] Center for Internet Security(CIS). 2024. https://www.cisecurity.org/. [Online; accessed 12-march-2024].

[10] Center for Information Security (CIS). 2024. CIS Kubernetes Benchmarks. https://www.cisecurity.org/benchmark/kubernetes

[11] Mubin Ul Haque, M Mehdi Kholoosi, and M Ali Babar. 2022. Kgsecconfig: a knowledge graph based approach for secured container orchestrator configuration. In *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 420–431.

[12] Sahand Hariri and Matias Carrasco Kind. 2018. Batch and Online Anomaly Detection for Scientific Applications in a Kubernetes Environment. In *Proceedings of the 9th Workshop on Scientific Cloud Computing* (Tempe, AZ, USA) *(Science-Cloud'18)*. Association for Computing Machinery, New York, NY, USA, Article 3, 7 pages. https://doi.org/10.1145/3217880.3217883

[13] Kacper Kamieniarz and Wojciech Mazurczyk. 2024. A Comparative Study on the Security of Kubernetes Deployments. In *2024 International Wireless Communications and Mobile Computing (IWCMC)*. 0718–0723. https://doi.org/10.1109/IWCMC61514.2024.10592468

[14] Kubebench. 2024. Kubebench. https://aquasecurity.github.io/kube-bench/v0.6.15/. [Online; accessed 15-March-2024].

[15] Kubebench. 2025. Kubebench-eks. https://github.com/aquasecurity/kube-bench/tree/main/cfg/eks-1.2.0. [Online; accessed 15-January-2025].

[16] Kubernetes. 2021. Production-Grade Container Orchestration. https://kubernetes.io/

[17] Kubescape. 2024. Kubescape. https://hub.armosec.io/docs/controls. [Online; accessed 15-March-2024].

[18] Kubescape. 2025. k8s-interface. https://github.com/kubescape/k8s-interface/. [Online; accessed 15-January-2025].

[19] Kaixuan Li, Yue Xue, Sen Chen, Han Liu, Kairan Sun, Ming Hu, Haijun Wang, Yang Liu, and Yixiang Chen. 2024. Static Application Security Testing (SAST) Tools for Smart Contracts: How Far Are We? *arXiv preprint arXiv:2404.18186* (2024).

[20] Andrew Martin and Michael Hausenblas. 2021. *Hacking Kubernetes: Threat-Driven Analysis and Defense*. O'Reilly Media.

[21] S. Miles. 2020. *Kubernetes: A Step-By-Step Guide For Beginners To Build, Manage, Develop, and Intelligently Deploy Applications By Using Kubernetes (2020 Edition)*. Independently Published. https://books.google.com/books?id=M4VmzQEACAAJ

[22] Akond Rahman, Shazibul Islam Shamim, Dibyendu Brinto Bose, and Rahul Pandita. 2023. Security Misconfigurations in Open Source Kubernetes Manifests: An Empirical Study. *ACM Trans. Softw. Eng. Methodol.* 32, 4, Article 99 (may 2023), 36 pages. https://doi.org/10.1145/3579639

[23] RedHat. 2024. The state of Kubernetes security report: 2024 edition. https://www.redhat.com/en/engage/state-kubernetes-security-report-2024

[24] Johnny Saldana. 2015. *The Coding Manual for Qualitative Researchers*. SAGE.

[25] Shazibul Islam Shamim, Hanyang Hu, and Akond Rahman. 2025. On Prescription or Off Prescription? An Empirical Study of Community-prescribed Security Configurations for Kubernetes. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering* (Ottawa, Canada) *(ICSE '25)*. Association for Computing Machinery, New York, NY, USA, 13 pages.

[26] Chin-Wei Tien, Tse-Yung Huang, Chia-Wei Tien, Ting-Chun Huang, and Sy-Yen Kuo. 2019. Kubanomaly: anomaly detection for the docker orchestration platform with neural network approaches. *Engineering reports* 1, 5 (2019), e12080.

[27] Trivy. 2024. Trivy. https://aquasecurity.github.io/trivy/. [Online; accessed 15-March-2024].

[28] Trivy. 2025. trivy-kubernetes. https://github.com/aquasecurity/trivy-kubernetes. [Online; accessed 15-January-2025].

[29] Hui Zhu and Christian Gehrmann. 2022. Kub-Sec, an automatic Kubernetes cluster AppArmor profile generation engine. In *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*. IEEE, 129–137.

[30] Aaron Zimba, Zhaoshun Wang, Mwenge Mulenga, and Nickson Herbert Odongo. 2018. Crypto mining attacks in information systems: An emerging threat to cyber security. *Journal of Computer Information Systems* (2018).