XI Commandments of Kubernetes Security: A Systematization of Knowledge Related to Kubernetes Security Practices

Md. Shazibul Islam Shamim
Dept. of Computer Science
Tennessee Technological University
Cookeville, TN, USA
mshamim42@students.tntech.edu

Farzana Ahamed Bhuiyan
Dept. of Computer Science
Tennessee Technological University
Cookeville, TN, USA
fbhuiyan42@students.tntech.edu

Akond Rahman
Dept. of Computer Science
Tennessee Tech. University
Cookeville, TN, USA
arahman@tntech.edu

***Abstract*—Kubernetes is an open-source software for automating management of computerized services. Organizations, such as IBM, Capital One and Adidas use Kubernetes to deploy and manage their containers, and have reported benefits related to deployment frequency. Despite reported benefits, Kubernetes deployments are susceptible to security vulnerabilities, such as those that occurred at Tesla in 2018. A systematization of Kubernetes security practices can help practitioners mitigate vulnerabilities in their Kubernetes deployments. *The goal of this paper is to help practitioners in securing their Kubernetes installations through a systematization of knowledge related to Kubernetes security practices.* We systematize knowledge by applying qualitative analysis on 104 Internet artifacts. We identify 11 security practices that include (i) implementation of role-based access control (RBAC) authorization to provide least privilege, (ii) applying security patches to keep Kubernetes updated, and (iii) implementing pod and network specific security policies.**

***Index Terms*—containers, devops, devsecops, grey literature, kubernetes, practices, review, security, systematization of knowledge**

I. Introduction

Kubernetes is an open-source software for automating management of computerized services, such as containers [1]. Practitioners use Kubernetes because it reduces repetitive manual processes involved in container deployment and management. Kubernetes is considered one of the most popular open-source container orchestration tools and it is used in organizations such as Adidas, Nokia, Spotify, and the U.S. Department of Defense (DoD) [2], [3]. Benefits of Kubernetes usage have been documented: for example usage of Kubernetes in the U.S. DoD resulted in reducing an eight month software deployment effort down to one week [3]. For Adidas, the load time for an e-commerce website was reduced by half, and release frequency increased from once every 4∼6 weeks to 3∼4 times a day [2].

Despite reported benefits, Kubernetes users have reported their concerns related to Kubernetes security. The Cloud Native Computing Foundation conducted a survey with 1,337 practitioners and reported 40% of the survey participants to be concerned with Kubernetes security [4]. Anecdotal evidence supports practitioner-reported concerns related to Kubernetes security. For example, in 2018, malicious users gained access to Tesla's Amazon Web Services (AWS) resources using an insecure Kubernetes console [5].

Systematizing available knowledge regarding Kubernetes security practices could support practitioners in securing their Kubernetes installations. Such systematization of knowledge can be beneficial for practitioners who (i) want to understand what activities need to be executed to secure Kubernetes components and (ii) can use the derived list of practices as a benchmark to compare their state of security practices.

Systematization of knowledge can be conducted by analyzing Internet artifacts, such as blog posts and video presentations. Instead of academic forums, such as research conferences, practitioners often report what practices they use in Internet artifacts [6], [7]. In prior work, researchers have acknowledged the value of Internet artifacts in deriving practices, and analyzed Internet artifacts to summarize security practices used in DevOps [8], practices used for continuous deployment [9], and testing practices [6]. Analysis of Internet artifacts can be useful for systematizing Kubernetes security knowledge—a research topic that remains under explored [10]. By systematically analyzing Internet artifacts related to Kubernetes security we hypothesize to derive a list of security practices.

The goal of this paper is to help practitioners in securing their Kubernetes installations through a systematization of knowledge related to Kubernetes security practices.

We answer the following research question: ***RQ:* What Kubernetes security practices are reported by practitioners?**

We systematize knowledge related to Kubernetes security by conducting a grey literature review [11] where we apply qualitative analysis on Internet artifacts. We collect required Internet artifacts using the Google search engine with three search strings. Next, we apply a set of filtering criteria and apply qualitative analysis [12] on 104 Internet artifacts, such as blog posts, to construct a list of security practices.

We list our contributions as following:

- A synthesized list of security practices for Kubernetes; and

978-1-7281-8388-6/20/$31.00 ©2020 IEEE
DOI 10.1109/SecDev45635.2020.00025

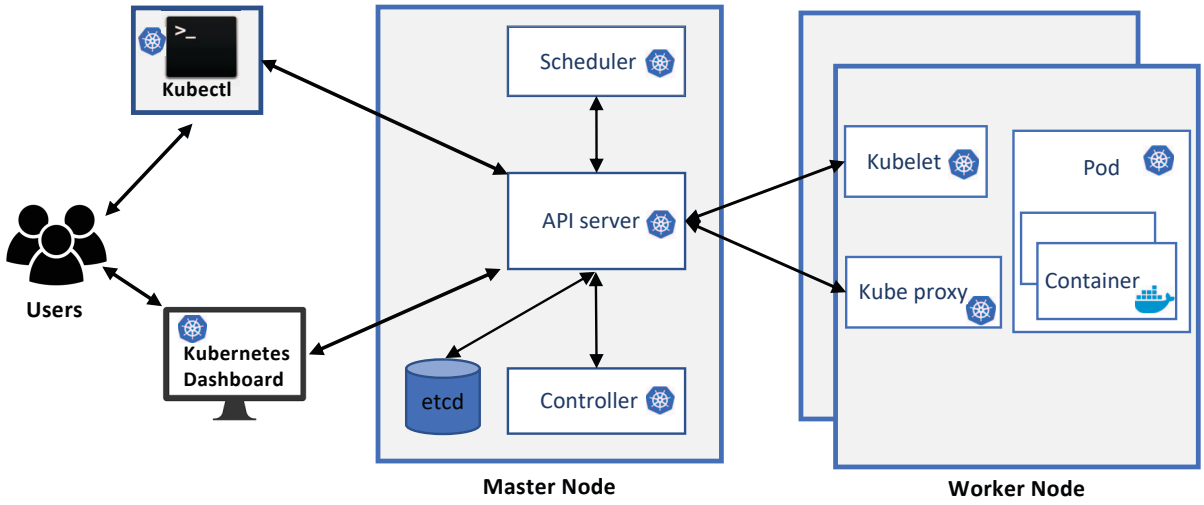

Fig. 1: A brief overview of Kubernetes. Kubernetes users interact with the installation using the Kubernetes dashboard and 'kubectl'. The purpose of master node is to maintain the desired cluster state and manage worker nodes. Worker nodes are used to run containerized applications inside the pod.

- A curated dataset [13] with a mapping between Internet artifact and identified security practices.

The rest of the paper is organized as follows: in Section II, we provide methodology of our paper. In Section III, we describe our derived Kubernetes security practices in details. In Section V, we discuss prior research on Kubernetes. We discuss our findings with implication for users and researchers and conclude the paper in Section VI.

II. Methodology

We first provide background on Kubernetes in Section II-A. Next, we provide methodology details in Section II-B.

A. Background

Kubernetes is an open-source software for automating management of computerized services such as containers [1]. A Kubernetes installation is colloquially referred to as a Kubernetes cluster [1]. Each Kubernetes cluster contains a set of worker machines defined as nodes. As shown in Figure 1, two types of nodes exist for Kubernetes: master nodes and worker nodes.

Each master node includes the following components: 'API server', 'scheduler', 'controller', and 'etcd' [1]. The 'API server' is responsible for orchestrating all the operations within the cluster. Kubernetes serves its functionality through an application program interface from the 'API server'. The 'controller' is a component on the master that watches the state of the cluster through the 'API server' and changes the current state towards the desired state. The 'scheduler' is the component in the control plane responsible for scheduling pods across multiple nodes. The 'etcd' is a key-value based database that stores all configuration information for the Kubernetes cluster. Users use a command-line tool 'Kubectl' to communicate with the 'API server' in the master node.

The worker nodes host the applications that run on Kubernetes [1]. The following components are included in the worker node: 'kube-proxy', 'kubelet' and 'pod'. 'kube-proxy' maintains the network rules on nodes. 'kubelet' is an agent that ensures containers are running inside a 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 associated dependencies to run in any computing environment [1].

B. Methodology to Identify Kubernetes Security Practices

We synthesize Kubernetes security practices by conducting a grey literature review [11]. A grey literature review is the process of reviewing and synthesizing content included in Internet artifacts, such as blog posts and video presentations [11]. A grey literature review is different from a systematic mapping study or systematic literature review, as in these types of literature reviews, researchers use peer-reviewed scientific articles indexed in scholar databases. In prior work, researchers have reported that practitioners use Internet artifacts, such as blog posts to report their experiences, recommendations, and the practices they follow. Previously,

researchers have systematically studied Internet artifacts to identify challenges in microservices development, identify practices used in continuous deployment [9], identify security practices used in organization who have adopted DevOps [8], and software testing [14]. Our hypothesis is that by systematically analyzing Internet artifacts we can synthesize Kubernetes security practices reported by practitioners.

We conduct grey literature review using the following steps: ***Step#*I-Collect Internet Artifacts**: We use the Google search engine to collect our Internet artifacts. We use 3 search strings: 'kubernetes security practices', 'kubernetes security good practices', and 'kubernetes security best practices'. We start with the search string 'kubernetes security practices', and later on added the other 2 search strings because while collecting search results with the first string we observe practices being referred to as 'good practices' and 'best practices'. After performing the search we collect the first 100 search results, as Google displays the results in a sorted order based on relevance.

***Step#*II-Select Internet Artifacts**: We apply an inclusion criteria on the collected search results to identify Internet artifacts that discuss security practices for Kubernetes. The inclusion criteria is listed below:

- The Internet artifact is not a duplicate;
- The Internet artifact is available for reading; and
- The Internet artifact discusses security practices for Kubernetes;

***Step#*III-Qualitative Analysis**: We use open coding [12], a qualitative analysis technique, to determine the security practices for Kubernetes. In open coding a rater observes and synthesizes patterns within unstructured text [12]. To determine the practices, the first author apply open coding on the content of the Internet artifacts to derive the security practices. The first author is a graduate student with a professional experience of one year in Kubernetes, and one year of academic experience in software security.

***Step#*IV-Verify Rating**: The process of determining the practices is susceptible to first author bias. We mitigate this bias by allocating another rater, the second author of the paper, who apply closed coding [15] on a randomly selected set of 50 Internet artifacts. Closed coding is the technique of mapping an entry to a pre-defined category [15]. For each of the 50 Internet artifacts, the second author examined if the artifact of interest includes a discussion related to the security practices identified by the first author. The second author has 3 years of experience in software security. We calculate the agreement rate between the first and second author for the 50 Internet artifacts using Cohen's Kappa [16].

III. Kubernetes Security Practices

In this section we answer: ***RQ: What Kubernetes security practices are reported by practitioners?*** After applying open coding on 104 Internet artifacts we derive 11 practices for Kubernetes security. Of the 104 Internet artifacts 90.38%, 4.81%, and 4.81% are respectively blog posts, videos and presentations. We describe each of these practices below, where the count of Internet artifacts is enclosed within parenthesis:

I. **Authentication and Authorization (82)**: The practice of applying authentication and authorization rules to prevent malicious users from getting access and performing unauthorized activities inside the Kubernetes cluster. Authentication in Kubernetes refers to the authentication of API requests through authentication plugins [17]. Authorization in Kubernetes refers to the evaluation of each authenticated API request against all policies to allow or deny the request [17]. Practitioners have reported a set of tasks to implement the practice of authentication and authorization:

- Anonymous access to the Kubernetes server needs to be disabled. By default, Kubernetes allows anonymous access to the Kubernetes API server. [17]
- Default authorization modes need to be disabled.
- Admission controllers need to be enabled. In Kubernetes, an admission controller is a tool that intercepts requests to the Kubernetes API after the request is authenticated and authorized, but before a volume is made persistent.
- Controlling the use of impersonation: Kubernetes allows one user to act as another user through impersonation headers [17]. The impersonation feature has benefits, for example, a user designated as an admin can use this feature to debug authorization by impersonating another user and checking if the request was denied. However, in case of failure to define limitations on who can impersonate and what the impersonated user can do, the impersonation feature can be detrimental to the security of Kubernetes.
- Default configurations must be changed. The use of default configuration in authentication and authorization can allow any anonymous unauthenticated user to perform malicious activities. For example, a malicious user can guess the default configuration of an insecure admission, gain access to the admission controller, and run malicious commands.

For authentication and authorization, practitioners suggest the use of OpenID [1], a standard protocol for authentication. The official Kubernetes documentation also provides guidelines on how to implement secure authorization using webhooks, role-based access control (RBAC) and attribute-based access control (ABAC) [17].

II. **Implementing Kubernetes-specific Security Policies (81):** The practice of applying policies to secure Kubernetes components, pods and network of Kubernetes clusters to prevent security breaches.

- *Network-specific policies*: The practice of applying a network policy to protect communication between Kubernetes pods from undesirable network communications. By default, all Kubernetes pods can communicate with other pods. Practitioners recommend policies to restrict traffic between pods, restrict API server access and reducing network exposure to secure the network. If network policies are not defined and firewalls are not set, then anyone may attack the API server from any IP address. Practitioners also suggest

[1]https://openid.net/

imposing proper firewalls to block all undesirable network communication using network policy plugins like Calico [2] and configuring restricted access to a database for pods.

- *Pod-specific policies*: The practice of implementing a policy for pods to apply security context to pods and containers. Pod policies determine how the workloads should run in the Kubernetes cluster. Without defining a secure context for the pod, a container may run with root privilege and write permission into the root file system, which can make the Kubernetes cluster vulnerable. Practitioners recommend containers inside a pod must run as a non-root user with read-only permission and enabling Linux security modules. Practitioners also recommend that users install the minimal version of operating systems to reduce the attack surface.
- *Generic policies*: The practice of applying a generic security policy to protect Kubernetes cluster components from external malicious users. TCP ports for kubelet, API server, etcd, and network plugins should not be left open and should require authentication to have visibility. Every user in the system should have the least privilege by default. Public SSH access to Kubernetes cluster nodes should be restricted. Practitioners recommend that Kubernetes users create an audit policy for logging, and audit policies must be configured for each Kubernetes cluster at the API server level.

III. **Vulnerability scanning (63):** The practice of scanning Kubernetes components and continuous delivery (CD) components for vulnerabilities.

- Kubernetes components, such as containers can contain vulnerabilities and malicious malware. If vulnerabilities are present in a Kubernetes cluster, then the entire container orchestration system, and the provisioned applications, become susceptible to attacks. For example, in 2017, researchers found Docker images embedded with malicious malware. Practitioners recommended scanning containers for vulnerabilities with tools,such as 'Dockscan' [3] and 'CoreOS Clair' [4].
- If images and deployment configurations within CD components are not inspected, then it can make the Kubernetes cluster vulnerable to malicious users. The malicious users can gain access at a later point when these images are deployed and may exploit the latent vulnerabilities in Kubernetes production environments. Practitioners recommend pulling images from a trusted private registry and checking for the vulnerability of code and images.

IV. **Logging (47):** The practice of enabling and monitoring logs for the Kubernetes cluster. Practitioners recommend that logging should be enabled for (i) applications, (ii) the containers within each pod, and for (iii) Kubernetes clusters for system health checking. Without enabling logging and monitoring, users may face difficulty troubleshooting unexpected consequences, such as attacks from malicious users and outages. To implement the practice of logging, practitioners propose the following practices:

- Logs must be monitored at a regular interval.
- Alerts must be set up for any drastic change in log metrics comparing to previous log records.

V. **Namespace separation (36):** The practice of separating namespaces so that the resource of one namespace are not shared with another. A 'namespace' in Kubernetes is a logically isolated virtual cluster within the same physical cluster. [17] Creation of separate namespaces enables resources to be isolated between namespaces. If a separate namespace is not created for a resource then the resource gets 'default' namespace. Practitioners recommend that each team in a company should have a separate namespace for better manageability and running its development and production environments[5]. If there is only a 'default' namespace and no separate namespace for different teams then any malicious user can perform an attack on the 'default' namespace making the entire resource vulnerable to that attack. Practitioners use the *–namespace* flag in kubectl command to separate namepsaces.

VI. **Encrypt and restrict access to etcd (34):** The practice of encrypting and restricting access to 'etcd', the internal database used by Kubernetes [17]. Practitioners recommend 'etcd' to only be available from the API servers, and to be isolated behind a firewall so that outsiders can not get access via API.

By default, Kubernetes stores secret data as plaintext in 'etcd'[6]. In that case, if a malicious user gets access to 'etcd', then the malicious user can retrieve sensitive information, such as database user names, passwords, and queries. Although Kubernetes does encrypt 'etcd', the key for the encryption is stored as plaintext in the config file in the master node. For that reason, practitioners recommend using secret management tools for additional security [17], such as 'Vault'[7] for encryption.

VII. **Continuous update (28):** The practice of applying security patches to keep the Kubernetes cluster updated with latest security fixes. Practitioners recommend that Kubernetes users apply updates as well as conducting continuous updates for the deployed applications within the Kubernetes pods. Without continuous updates, vulnerabilities might exist in the Kubernetes installation, which can give malicious users opportunity to perform attacks.

Vulnerabilities in Kubernetes are not uncommon: for example, two vulnerabilities CVE-2019-16276 [18] and CVE-2019-11253 [19] were discovered in October 2019 in Kubernetes [8]. The vulnerability 'CVE-2019-16276' was related to 'CWE-444 Inconsistent Interpretation of HTTP Requests ('HTTP Request Smuggling')'. The vulnerability 'CVE-2019-11253' was related to 'CWE-20 Improper Input Validation'.

[2]https://www.projectcalico.org/calico-networking-for-kubernetes/
[3]https://github.com/kost/dockscan
[4]https://github.com/quay/clair
[5]https://cloud.google.com/blog/products/gcp/kubernetes-best-practices-organizing-with-namespaces
[6]https://ubuntu.com/kubernetes/docs/encryption-at-rest
[7]https://www.vaultproject.io
[8]https://security.berkeley.edu/news/kubernetes-vulnerabilities-allow-authentication-bypass-dos-cve-2019-16276

The security patches for 'CVE-2019-11253' and 'CVE-2019-16276' were released on October 16, 2019 and October 22, 2019 respectively [9]. If any Kubernetes user does not install these security patches then the Kubernetes cluster will be susceptible to a denial of service attack.

For continuous updates, practitioners have also recommended the use of rolling update, i.e. installing Kubernetes patches without disrupting the availability of the deployed applications.[10] Kubernetes provides tools, such as 'kubectl' to perform rolling updates [17].

VIII. **Limit CPU and memory quota (18):** The practice of limiting CPU and memory to a pod or a namespace so that malicious attacks can be mitigated. By default, all resources in Kubernetes start with unbounded memory requests/limits and unbounded CPU access. If a malicious user starts a denial of service (DOS) attack with in a pod within the Kubernetes cluster then, due to a high volume of requests, kube-scheduler will create a new pod and an instance of the container will start inside the new pod. This process continue until it consumes all available CPU resources and memory leaving all the applications in starvation. Hence, failure to define CPU and memory request limits for a pod or the namespace may result in a consumption of all available resources in the Kubernetes cluster, enabling a denial of service (DOS) attack.

Practitioners can configure the amount of resources by defining a maximum number of instances for a container, the number of CPU share for an application to consume, and the maximum amount of memory for a pod or namespace.

IX. **Enable SSL/TLS support (18):** The practice of enabling secure sockets layer (SSL) or transport layer security (TLS) protocol to ensure secure and encrypted communication between Kubernetes components. Enabling TLS between kubernetes api server, etcd, kubelet and kubectl ensures secure communication between cluster components. Practitioners suggest enabling TLS and SSL certificates for Kubernetes components.

X. **Separate sensitive workload (14):** The practice of running sensitive applications on a dedicated set of machines to limit the potential impact of a security breach. For example, if a malicious user gets access to a node's 'kubelet' credentials, then the user can access the contents of secrets and gain control of the entire file system, but the user will not be able to access the sensitive applications and associated secrets. Practitioners recommend Kubernetes-provided utilities, such as 'taints and tolerations' [17] that can control where a pod might be deployed.

XI. **Secure metadata access (9):** The practice of securing the sensitive metadata of the Kubernetes cluster. Practitioners state that the Kubernetes metadata APIs provide a gateway to expose 'kubelet' admin credentials. Google recommends activating features such as 'Workload Identity'[11] for Google Kubernetes Engine (GKE) to prevent any sensitive information from leaking through the metadata service.

[9]https://cloud.google.com/kubernetes-engine/docs/security-bulletins

[10]https://k8s.vmware.com/kubernetes-security-best-practices/

[11]https://cloud.google.com/kubernetes-engine/docs/how-to/protecting-cluster-metadata

Rater verification: The Cohen's Kappa between the two raters is 0.8, which is substantial according to Landis and Koch [20].

IV. Threats to Validity

We discuss the limitations of our paper as following:

Conclusion Validity: Our derived set of practices is limited to our collection of 104 Internet artifacts. Our collection of Internet artifacts might have missed Internet artifacts, that may have included practices not identified in our paper. We mitigate this limitation by systematically collecting a set of 104 Internet artifacts.

The identified practices are also susceptible to biases of the rater who identified the practices by applying open coding. We mitigate this limitation by allocating another rater, who applied closed coding. The Cohen's Kappa between the two raters is 0.8. which is substantial [20].

Construct Validity: Our identified categories are susceptible to experimenter bias. The first author who derived the practices has professional experience in Kubernetes. The first author's professional experience can formulate expectations related to security practices for Kubernetes, which may influence the identified practices.

External Validity: Our findings might not be generalizable as we might have excluded practices unique to the proprietary domains, and not discussed publicly in Internet artifacts.

V. Related Work

Our paper is related to prior research that has investigated usage and maintenance of Kubernetes. Burns et al. [21] described the evolution of container management systems at Google, and described how two initial internal systems called Borg and Omega was evolved into Kubernetes. Brewer [22] conducted a case study on Kubernetes and discussed how key concepts of Kubernetes can be used to simplify scaling of containers. Medel et al. [23] used real data collected from Kubernetes and applied formal modeling to characterize performance and resource management in Kubernetes. Chang et al. [24] constructed a monitoring platform to dynamically provision cloud resources using Kubernetes. Vayghan et al. [25] investigated availability of Kubernetes using a set of experiments, and reported that service outages can occur frequently. Shah and Dubaria [26] compared orchestration management features of Docker Swarm, Kubernetes, and Google Cloud Platform, and observed Kubernetes to provide features, such as deployment, monitoring, and easy scalability. Takahashi et al. [27] proposed a portable load balancer for Kubernetes, and reported improved portability without sacrificing performance. Song et al. [28] used Kubernetes to construct an auto scaling system for API gateways. The authors [28] report that their constructed system improves utilization of system resources, while ensuring high availability. Muralidharan et al. [29] constructed a Kubernetes-based system to monitor and manage Internet of Things (IoT) applications for smart cities. Weiguo et al. [30] constructed a resource scheduling algorithm for Kubernetes using ant colony and particle swarm optimization

techniques. The scheduling algorithm proposed by Wei-guo et al. [30] outperforms the original algorithm used in Kubernetes.

The above-mentioned discussion highlights Kubernetes research in two areas: (i) use of Kubernetes in creating systems, such as monitoring systems and (ii) case studies on Kubernetes related to performance and resource management. We observe a lack of research related to security practices for Kubernetes. We address this research gap by systematically synthesizing practitioner-reported security practices using grey literature review.

VI. Discussion and Conclusion

With great power comes responsibilities: Kubernetes provide utilities for users to manage containers at scale. However, our description of the 11 practices in Section III shows that effective and secure usage of Kubernetes requires the implementation of security practices applicable for multiple components within the Kubernetes installations: containers, pods, 'etcd' database etc. The application of the aforementioned 11 practices also need a deep understanding of Kubernetes components and configurations. Our discussion in Section III can be helpful in two ways: *first*, understand the components where security practices are applicable. *Second*, practitioners who already have Kubernetes in place, can use our identified practices as a benchmark and compare their usage of practices.

Implication for researchers: Our discussion in Section V shows that Kubernetes security to be an under explored research area. Our derived list of security practices can provide the groundwork for future research in Kubernetes security. To what extent the reported security practices are in use can be quantified systematically. Researchers can measure the attack surface associated with Kubernetes components and configurations. Researchers can find possible mitigation strategies such as static analysis and dynamic analysis to inspect insecure practices in Kubernetes. Researchers can also quantify how frequently the identified practices are actually in use. Furthermore, detection and mitigation of security misconfigurations that occur in Kubernetes could be of interest to researchers.

Conclusion: As Kubernetes usage becomes increasingly popular, securing Kubernetes is of paramount importance to practitioners. A systematization of knowledge related to practitioner-reported practices might be helpful to secure Kubernetes installations. We conduct a qualitative analysis of Internet artifacts, such as blog posts to identify 11 security practices for Kubernetes. Our derived list of practices include continuous update, enable SSL/TLS support, vulnerability scanning and logging. Our paper can help practitioners in securing Kubernetes installations. Further, our findings can lay the groundwork to conduct research in Kubernetes security.

Acknowledgement

We thank the members of the PASER research group at Tennessee Technological University and Patrick Morrison from IBM for their valuable feedback.

References

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

[2] Kubernetes User Case Studies, May 2020. [Online]. Available: https://kubernetes.io/case-studies/

[3] With Kubernetes, the U.S. Department of Defense Is Enabling DevSecOps on F-16s and Battleships, May 2020. [Online]. Available: https://www.cncf.io/case-study/dod/

[4] CNCF SURVEY 2019, March 2019. [Online]. Available: https://www.cncf.io/wp-content/uploads/2020/03/CNCF_Survey_Report.pdf

[5] Tesla cloud resources are hacked to run cryptocurrency-mining malware, February 2018. [Online]. Available: https://arstechnica.com/information-technology/2018/02/tesla-cloud-resources-are-hacked-to-run-cryptocurrency-mining-malware/

[6] V. Garousi and B. Küçük, "Smells in software test code: A survey of knowledge in industry and academia," *Journal of Systems and Software*, vol. 138, pp. 52–81, 2018.

[7] R. L. Glass, *Software Creativity 2.0*. developer.* Books, 2006.

[8] A. A. Ur Rahman and L. Williams, "Software security in devops: Synthesizing practitioners' perceptions and practices," in *Proceedings of the International Workshop on Continuous Software Evolution and Delivery*, ser. CSED '16. New York, NY, USA: ACM, 2016, pp. 70–76. [Online]. Available: http://doi.acm.org/10.1145/2896941.2896946

[9] A. A. U. Rahman, E. Helms, L. Williams, and C. Parnin, "Synthesizing continuous deployment practices used in software development," in *Proceedings of the 2015 Agile Conference*, ser. AGILE '15. USA: IEEE Computer Society, 2015, p. 1–10. [Online]. Available: https://doi.org/10.1109/Agile.2015.12

[10] Advanced Persistence Threats - The Future of Kubernetes Attacks, March 2020. [Online]. Available: https://darkbit.io/blog/future-kubernetes-attacks-rsa-2020

[11] S. Hopewell, M. Clarke, and S. Mallett, "Grey literature and systematic reviews," *Publication bias in meta-analysis: Prevention, assessment and adjustments*, pp. 49–72, 2005.

[12] J. Saldaña, *The coding manual for qualitative researchers*. Sage, 2015.

[13] Anonymous, "Dataset for paper." [Online]. Available: https://figshare.com/s/548f0f90a0f2744cf33a

[14] V. Garousi, M. Felderer, and T. Hacaloğlu, "Software test maturity assessment and test process improvement: A multivocal literature review," *Information and Software Technology*, vol. 85, pp. 16 – 42, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584917300162

[15] B. F. Crabtree and W. L. Miller, *Doing qualitative research*. sage publications, 1999.

[16] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960. [Online]. Available: http://dx.doi.org/10.1177/001316446002000104

[17] Kubernetes, "Production-grade container orchestration." [Online]. Available: https://kubernetes.io/docs/

[18] National Vulnerability Database, September 2019, [online] https://nvd.nist.gov/vuln/detail/CVE-2019-16276. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2019-16276

[19] National Vulnerability Database, October 2019, [online] https://nvd.nist.gov/vuln/detail/CVE-2019-11253. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2019-11253

[20] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977. [Online]. Available: http://www.jstor.org/stable/2529310

[21] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, omega, and kubernetes," *Queue*, vol. 14, no. 1, p. 70–93, Jan. 2016. [Online]. Available: https://doi.org/10.1145/2898442.2898444

[22] E. A. Brewer, "Kubernetes and the path to cloud native," in *Proceedings of the Sixth ACM Symposium on Cloud Computing*, ser. SoCC '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 167. [Online]. Available: https://doi.org/10.1145/2806777.2809955

[23] V. Medel, O. Rana, J. a. Banares, and U. Arronategui, "Modelling performance & resource management in kubernetes," in *Proceedings of the 9th International Conference on Utility and Cloud Computing*, ser. UCC '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 257–262. [Online]. Available: https://doi.org/10.1145/2996890.3007869

[24] C. Chang, S. Yang, E. Yeh, P. Lin, and J. Jeng, "A kubernetes-based monitoring platform for dynamic cloud resource provisioning," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–6.

[25] L. Abdollahi Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "Deploying microservice based applications with kubernetes: Experiments and lessons learned," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, 2018, pp. 970–973.

[26] J. Shah and D. Dubaria, "Building modern clouds: Using docker, kubernetes google cloud platform," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 2019, pp. 0184–0189.

[27] K. Takahashi, K. Aida, T. Tanjo, and J. Sun, "A portable load balancer for kubernetes cluster," in *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, ser. HPC Asia 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 222–231. [Online]. Available: https://doi.org/10.1145/3149457.3149473

[28] M. Song, C. Zhang, and E. Haihong, "An auto scaling system for api gateway based on kubernetes," in *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, 2018, pp. 109–112.

[29] S. Muralidharan, G. Song, and H. Ko, "Monitoring and managing iot applications in smart cities using kubernetes," *CLOUD COMPUTING*, vol. 11, 2019.

[30] Z. Wei-guo, M. Xi-lin, and Z. Jin-zhong, "Research on kubernetes' resource scheduling scheme," in *Proceedings of the 8th International Conference on Communication and Network Security*, ser. ICCNS 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 144–148. [Online]. Available: https://doi.org/10.1145/3290480.3290507