Investigation into Kubernetes Security – Deficiencies and Best Practices

Filip Sannervik

May 31, 2022

1 Introduction

As our society is heading into an ever increasing digital age, organisations within the industry are moving towards container-based infrastructure for managing their workloads and deploying their software services. This is known as serverless computing or Function-as-a-service (Faas), which allows developers to deploy functions and digital units without worrying about the underlying infrastructure [16]. Currently serverless solutions are generally available at most cloud service providers. It has many reported benefits such as reducing costs, improving scalability, and reduced developer concern about infrastructure.

Kubernetes is an open-source container orchestration tool used by many large organisations such as IBM, Spotify, Huwaei, and Adidas to handle their containers [1]. There are a lot of reported benefits with using Kubernetes such as being able to more frequently deploy software and automated management of online infrastructure. Even though there are benefits, Kubernetes installations have been susceptible to security defects. An example of this is the 2018 Tesla breach where a Kubernetes console belonging to Tesla was not password protected, and in addition to this, one of the containers handled by the console included login information for a Tesla AWS cloud environment. Attackers then used this to gain access and deploy code to mine cryptocurrency on their systems [18]. This is just one example of many where vulnerabilities in Kubernetes installations has led to large-scale breaches, a similar incident happened to Jenkins servers around the same time.

As serverless architecture adoption is rising, with Kubernetes being the leading container-orchestration system [13], the security aspect of these solutions needs to be taken into close consideration. This essay aims to describe some common security vulnerabilities apparent in Kubernetes in order to increase awareness, and also describe some security practices that can greatly increase the security of Kubernetes deployments.

2 Background

2.1 Relevance of Kubernetes in DevOps

"Quality deliveries with short cycle time need a high degree of automation" [9], this alongside increasing demands for software that is more complex and shorter delivery times creates the need for a better process and automation [4].

DevOps solves this problem using the DevOps tools for building, deployment and testing using pipelines. The pipelines can automate the mentioned tasks as well as give a better cooperation between developers and operation teams. However, another problem that arises which needs to be solved is the dependency management in the pipelines. Once the need to update components within the code or adding dependencies arises it can create the problem of updating the pipelines. This is solved by containerization [14, 9].

A container image can be seen as a executable code that contains all necessities to run the underlying software. These necessities include dependencies, configuration files, etc [19, 15]. Since there can be multiple containers which are located at different geographic locations the need for a system that can manage the different containers becomes apparent. The solution is a container orchestration platform, such as Kubernetes [12].

2.2 The Kubernetes Environment

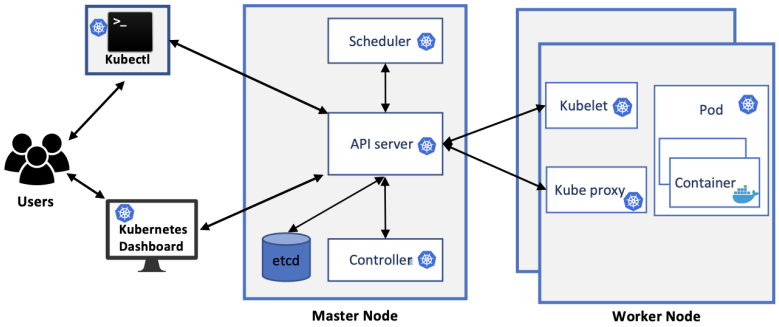

Figure 1: Brief overview of a Kubernetes installation. [10]

A Kubernetes cluster has at least one master node and at least one worker node. The master node's purpose is to maintain the desired state of the cluster as well as manage worker nodes, it is communicated with through the Kubectl console. Through the console users can run commands against Kubernetes clusters such as deploy applications, inspect and manage resources, as well as log viewing, among other things. The purpose of the worker nodes are to run containerized applications inside the pods. The API server is responsible for coordinating all aspects of a cluster, it accepts requests for updating components within the cluster. The Etcd stores all of Kubernetes data, since Kubernetes is a distributed system it needs a distributed data store. The Kubelet is an agent that acts as a bridge between the Kubernetes master and nodes. It receives pod specifications through the API server and makes sure the containers are working as they should [17, 10].

3 Security Deficiencies

A paper titled *'Under-reported' Security Defects in Kubernetes Manifests* researched how apparent security defects are in Kubernetes installations and found that on 5193 commits collected from 38 open-source repositories only 0.79% of the commits were security related [8]. The researchers found that security-related defects within Kubernetes installations are under-reported which highlights the importance of research within the area since the ever increasing use of Kubernetes and serverless solutions could lead to large-scale security breaches if left unresolved.

There are multiple attack vectors on a Kubernetes architecture. In this essay we will highlight some of the most important ones that has been used in previous high-profile attacks. As explained in the background, Kubernetes environments adds extra layers compared to traditional non-serverless infrastructure. This adds new attack vectors which require security measures.

3.1 Misconfigured Dashboard

The Kubernetes dashboard is a web-based user interface where users interact with the containers and cluster resources as shown in figure 1. It can be used to deploy and troubleshoot containerized applications, manage cluster resources, get an overview of applications running on your cluster, as well as creating or modifying Kubernetes resources [6].

It is important that the dashboard is behind strong protection so that it is not publicly accessible. To access the dashboard a kubeconfig file or a token should be necessary. The kubeconfig file is signed by the CA (certificate authority) of the cluster giving the owner access to the dashboard. For a kubeconfig file to be created, a user generates a key which is used to create a CSR (certificate signing request). Then administrators of the cluster receive the CSR file and imports it into the cluster. The CSR is then signed creating a CRT file (security certificate). The CRT file is used to generate a kubeconfig file and information like cluster name, user, and the CA. A role is also created that specifies the rights of the user inside the cluster. [11, 17]

If the dashboard is not configured correctly like explained above it could give attackers access to the cluster, which is what happened to Tesla in 2018. Attackers can then deploy pods with malicious functions. To minimize security deficiencies of the Kubernetes dashboard, on top of requiring a kubeconfig file to sign in, the dashboard should also be behind a firewall or reverse proxy.

3.2 Vulnerable Images

The most common adopted solution for containerized workloads is Kubernetes in combination with Docker. Kubernetes can however use other container runtimes than docker, for example Moby and Cri-o. Container runtimes are a critical part of the Kubernetes cluster, needing their own level of security apart from the general Kubernetes security. Images run by a container should be scanned for vulnerabilities before being stored in a trusted repository [17].

It is common that images are downloaded from public repositories such as docker hub. Since they are public it is possible that a user uploads an image with vulnerabilities. Large number of containers on docker hub are vulnerable both by intention and unintentionally. These vulnerabilities could provide unauthorized access to the containers and the host where docker (or other container runtime) is being run. Attackers can also backdoor a legitimate docker container, re-upload it to a public repository to trick organisations into downloading it if they mistake it for the legitimate version. Once the docker container is executed the attacker will gain access to the host [3, 17].

3.3 Privilege escalation

Privilege escalation is when an attacker escalates their access in a system within a security perimeter. An attacker might gain access to a system through a weak point where their access is limited outside the reach of sensitive systems or data. The attacker then uses this access to escalate their privileges gaining access more important systems [7].

An attacker that has gained access to a container, by a vulnerable image for example, can do privilege escalation and break out of it to reach the host and other parts of the Kubernetes cluster if there are security deficiencies such as misconfigurations or vulnerabilities. Container runtimes such as Docker are susceptible to privilege escalation vulnerabilities for different reasons, for example docker daemon requires root privileges to perform some of its operations. Also because it is more time-consuming and harder to configure applications to run on low privileges, it is common for containers deployed into production to be configured to run as root. Therefore misconfigurations such as running containers with the `--privileged` flag are very common. The privileged flag gives extra Linux capabilities to a container and more access to the systems resources it is running on. An attacker can use this extended access of the containers to escape the container and gain access to the host by loading modules into the Linux kernel. This means that vulnerabilities present within the Kubernetes cluster such as vulnerable images makes the whole orchestration system susceptible to attacks.

4 Kubernetes Security Practices

4.1 Vulnerability Scanning and Patching

One important practice to increase the security of Kubernetes installations is to continuously scan Kubernetes components and continuous delivery (CD) components for vulnerabilities. As mentioned

in the previous section, containers can contain vulnerabilities or malicious code, as the example with Docker images embedded with malware. Therefore scanning containers for vulnerabilities before sending them into production is highly important, there are tools that can be used for this, some popular ones are **Dockscan** and **Clair**. If images and configurations within CD components are not inspected the Kubernetes cluster might become vulnerable which can lead to malicious users exploiting these vulnerabilities in Kubernetes production environments. Another important practice to help avoid this is to only pull images from a trusted private registry or secure registered repositories. There are Kubernetes options that should be configured to implement policies in order to increase security, such as only allowing it to pull containers from secure repositories [10, 17].

Another practice that is important to secure a Kubernetes environment is to frequently apply security patches to keep the Kubernetes cluster up to date with the latest security fixes. Vulnerabilities in Kubernetes are continuously being discovered, and not doing regular updates will leave the vulnerabilities in the Kubernetes installation, giving malicious actors opportunity to perform attacks. Also because of the nature of containerized applications, upgrades will sanitize the cluster from compromised pods [10, 17].

4.2 Authentication and Authorization

We already touched on this topic in section 3.1 where we described how to securely configure the Kubernetes dashboard to prevent unauthorized access. The practice of correctly configuring authentication and authorization rules is one of the most important ones in securing a Kubernetes installation. When we talk about authentication in Kubernetes we mainly focus on the authentication of API requests, the API is usually accessed using the Kubectl console. Authorization focus on the evaluation of authenticated API requests to either allow or deny the request [5]. In this section we will describe important practices that implements sound authentication and authorization of a Kubernetes installation.

Kubernetes allows anonymous access to the API by default [5], this should be disabled in a securely configured installation since this access could be used for privilege escalation. Default authorization modes should be disabled. In Kubernetes there is a third step to the security of API requests known as Admission control, the three steps of API security in Kubernetes is shown in figure 2. The admission controller is the final step and handles requests after they have been authenticated and authorized, but before Kubernetes persists the resource. It limits the permissions of each Kubelet (see figure 1), making sure they can only modify pods located in its own node. It also ensures some commands from privileged containers are blocked [17]. Therefore it is an important security practice to have admission controllers enabled when configuring Kubernetes installations.

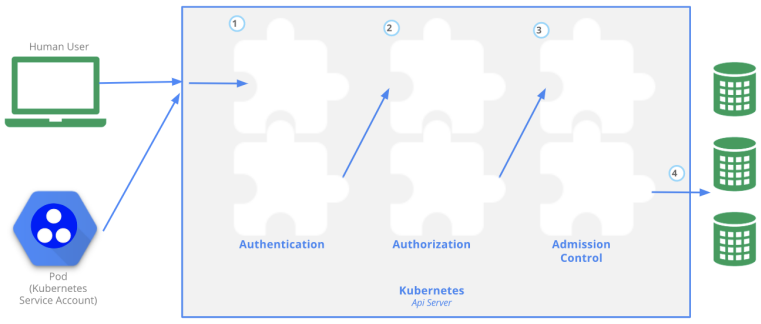

Figure 2: Kubernetes 3-step security before executing API requests [2].

5 Conclusion

A new paradigm is happening within the industry where practitioners are moving towards container-based infrastructure because of the many benefits it brings to software development. It is apparent

that adoption is moving towards serverless computing. With adoption of these systems increasing, the security of these systems cannot fall behind in order to avoid future large-scale breaches.

Kubernetes being the leading container orchestration tool, can have security deficiencies if not configured correctly. Therefore it is important for practitioners to be aware of the most important attack vectors of a Kubernetes installation so the correct steps can be conducted to secure their systems. To achieve this practitioners should regularly scan their components for vulnerabilities to avoid vulnerable or compromised containers to get into production. Updates should also be applied regularly to avoid leaving disclosed vulnerabilities on their systems, and to purge the cluster of infected containers. Authentication of dashboards and consoles used to interact with the Kubernetes cluster needs to be configured correctly, it is extremely important to not use the default configurations. Preferably use security certificates for optimal security. By following these practices organisations can reap all the benefits of serverless computing without worrying (as much) about security.

5.1 Reflection

Even though the topic is about security, I have learned a lot about Kubernetes as a container orchestration platform and serverless architecture as a whole. This being a very relevant topic in our time and probably one of the most central part of DevOps it has been a very educational experience. The topic within computer science that interests me the most is security and it is fun to see how I can learn about other aspects of computer science such as Kubernetes by leveraging my interest for security.

References

[1] Case studies. https://kubernetes.io/case-studies/. [Online; accessed 28-May-2022].

[2] Controlling access to the kubernetes api. https://kubernetes.io/docs/concepts/security/controlling-access/. [Online; accessed 31-May-2022].

[3] Docker hub. https://hub.docker.com/r/vulnerables/cve-2014-6271.

[4] The importance of kubernetes in devops. https://www.nexsoftsys.com/articles/importance-of-kubernetes-in-devops.html. [Online; accessed 30-May-2022].

[5] Kubernetes, "production-grade container orchestration.". https://kubernetes.io/docs/. [Online; accessed 31-May-2022].

[6] Deploy and access the kubernetes dashboard. https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/, Feb 2022.

[7] Understanding privilege escalation and 5 common attack techniques. https://www.cynet.com/network-attacks/privilege-escalation/#:~:text=Privilege%20escalation%20is%20a%20type,gaining%20access%20to%20a%20system., May 2022.

[8] D. B. Bose, A. Rahman, and S. I. Shamim. 'under-reported' security defects in kubernetes manifests. In *2021 IEEE/ACM 2nd International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)*, pages 9–12, 2021.

[9] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano. Devops. *IEEE Software*, 33(3):94–100, 2016.

[10] M. S. Islam Shamim, F. Ahamed Bhuiyan, and A. Rahman. Xi commandments of kubernetes security: A systematization of knowledge related to kubernetes security practices. In *2020 IEEE Secure Development (SecDev)*, pages 58–64, 2020.

[11] L. Juggery. Kubernetes tips: Give access to your cluster with a client certificate. https://betterprogramming.pub/k8s-tips-give-access-to-your-clusterwith-a-client-certificate-dfb3b71a76fe, Jan 2020.

[12] A. Khan. Key characteristics of a container orchestration platform to enable a modern application. *IEEE Cloud Computing*, 4(5):42–48, 2017.

[13] N. Matherson. 26 kubernetes statistics to reference, May 2022.

[14] G. D. Mayaan. Best of 2021 – devops and kubernetes: A perfect match? https://containerjournal.com/features/devops-and-kubernetes-a-perfect-match/, December 27 2018.

[15] A. Modak, S. D. Chaudhary, P. S. Paygude, and S. R. Ldate. Techniques to secure data on cloud: Docker swarm or kubernetes? In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 7–12, 2018.

[16] S. Mohanty et al. Evaluation of serverless computing frameworks based on kubernetes. Jul 2018.

[17] P. Mytilinakis. Attack methods and defenses on kubernetes. Master's thesis, Πανεπιστήμιο Πειραιώς, 2020.

[18] L. H. Newman. Hackers enlisted tesla's cloud to mine cryptocurrency. https://www.wired.com/story/cryptojacking-tesla-amazon-cloud/, Feb 2018.

[19] R. Powell. Benefits of containerization. https://circleci.com/blog/benefits-of-containerization/, September 15 2021.