Best practices for implementing Kubernetes security in production

Kubernetes is a container orchestration tool which is quite famous in a production environment as it has revolutionized the treatment of application deployment and how it should be managed. Though most of the organizations are moving towards Kubernetes as an orchestration solution, it is important to secure the cluster to minimize the risk of data leaks and security breaches. Kubernetes provides us many solutions which can be implemented to secure our cluster and here we will be discussing the best practices for implementing Kubernetes security in production.

1. **Implement Role-based Access Control (RBAC):**

We need a solution to authorize anyone to access the resources as per the requirements. Kubernetes introduced a mechanism called Role-based access control also known as RBAC. It is used to control the access or grant granular permissions to users,groups and service accounts. We can use this mechanism to control who has access to which resource in a cluster or in a namespace. However Kubernetes does not know how to create a user or a group in a cluster but we can give permission to them using the RBAC. Kubernetes supports service accounts as we can create, delete or integrate service accounts with the resources.

Let’s understand how RBAC should be implemented in a Kubernetes environment.

* Under the RBAC model, Kubernetes has introduced four objects which can be managed by RBAC: roles, role bindings, cluster roles and cluster role bindings.

Roles and role bindings are only applied to a specific namespace whereas cluster roles and cluster role bindings are applied to the entire cluster. Roles and cluster roles define the set of permission that will be granted to users, groups or service accounts, whereas role bindings and cluster role bindings link the roles or cluster roles to users, groups and service accounts.

* We should follow the principle of least privilege as each user or group or service account should be granted least privilege or minimum permission in order to perform the job. This way we are ensuring that there is not unnecessary permission given to anyone and our resources are secured. So we have to make sure to provide only those permissions which are required for each user or group or service account.
* Roles should be reviewed or updated on a regular basis which will ensure that they are committed to the company’s security policies. We should review a role considering if we are providing the necessary permissions to do the job and restricting it for any unnecessary privilege. If any changes are required, then we should update it accordingly. Review and update can be done as per the organization’s security policies.
* Kubernetes has this special object called service account that is used by pods to access the Kubernetes API. By default, we have a service account named default which is inherited by pods hence they are granted with full permission which is a security risk. We should provide the granular permission to a service account and link them using the RBAC objects.
* We should monitor and audit RBAC activity to ensure that it is being used as per organization’s security policies. We have tools like Prometheus or Grafana which can provide us the RBAC metrics like how many requests are denied which will give us an insight for further improvement.

1. **Implement Network Policies:**

By default, Kubernetes allows all network traffic between pods which is a security concern and it increases the risk of unauthorized access. Kubernetes provides a mechanism called Network Policy which is used to control the flow of network traffic to or from pods. This way we can improve the security of our cluster by creating some rules within a network policy. Let’s understand how we can implement Network policy.

* Before the implementation, we should understand the significance of a network policy. It is a Kubernetes object that defines a set of rules which are applied to a set of pods to control the network traffic.

We have two types of rules: Ingress and Egress. Ingress controls the incoming traffic whereas Egress controls the outgoing traffic. The network policy can be applied to a specific pod or all the pods in a namespace.

* A best practice is to implement a default deny policy which denies all the traffic to and from the pods unless you define rules to allow traffic explicitly. Hence this type of policy will prevent unauthorized access to our critical applications.
* Labels play an important role while implementing a network policy. We use pod’s labels which will group all the pods having the same label and the policy will track the pods having these labels to implement the rules automatically. This ensures that only the pods which want to communicate with each other are allowed and others are restricted to communicate.
* We can use Namespaces to limit the access and these namespace group the resources like pods, deployment, secrets etc. We can mention which namespace we want the pods to access or which resources within a namespace can be accessed. This will ensure that only authorized users or groups are permitted in a namespace.
* We should review and update the network policies on a regular basis just like RBAC. As per our organization’s security policies, we have to review the policies if there is any unauthorized access which is a security threat to our resources and update them accordingly.

1. **Secure Kubernetes API Server:**

Kubernetes API server is the main and central component in the cluster which helps the communication between the control plane and worker nodes. Hence we have to make sure that our API server is secure because this is the entrypoint to our cluster and it would impact the overall security of our cluster. Let’s understand how we can implement this.

* We should implement Transport Layer Security (TLS) which provides encryption and authentication hence we would be able to protect the communication between the Kubernetes API Server and its clients. We should configure Kubernetes API Server to use a valid SSL/TLS certificate from a trusted Certificate Authority (CA). For additional security, we can configure our Kubernetes API server to use client certificates for authentication.
* We can implement RBAC on our Kubernetes API Server to restrict the access so that only authorized users or groups can communicate with it. We should ensure that our API Server is restricted to public access and only trusted networks have the access to it.
* To enhance the security, we can use the Authentication methods which are supported by Kubernetes API Server. These methods are client certificates, static tokens and OAuth tokens. We can also use the combination of above methods for strong authentication like client certificates and OAuth tokens. Two factor authentication (2FA) can also be used for additional security.
* We should keep monitoring the Kubernetes API Server for suspicious activity or any unauthorized attempt using the API request. So it is recommended to log all API requests to help us identify any security threat.
* We should keep our Kubernetes API Server up to date so that we get a fix for any security vulnerability or bugs. This practice will ensure that we are free from any vulnerability and bugs hence our server will be secured.

1. **Secure Container Images:**

Every node in our cluster is having a container runtime engine which is responsible for pulling the image and running a container. If we pull the vulnerable images then it would be a huge security concern. Let’s understand how we can implement this in our cluster.

* We have to ensure that the images which are being used in our cluster are from trusted sources. We have public and private repositories like Docker Hub, Amazon Elastic Container Registry (ECR) etc which are used to store the images. So before pulling any image, we need to check if we are using the safe and secure repository.
* We can check for any security vulnerability or configuration issue using the image scanning tools. We can use tools like Aqua Security , Clair, Anchore etc to scan the container image before using them in a pod’s or deployment’s specification.
* To make sure if we are using an authentic image, we can use the image signing process which signs any image digitally and helps us to verify that the images are from an authenticated source. It also ensures that the images are not tempered. We have few tools like Sigstore and Notary to sign the images and verify the signature before the deployment.
* We can use ImagePolicyWebhook which defines some policies to check if the deployment is using the image from a trusted source and it is meeting our security requirements. For example, we can set a policy to pull the image from a specific repository only, else do not deploy with the untrusted image.
* We should keep our images up to date to fix any bugs or security vulnerabilities. We should always look for the latest security patches or bug fixes to secure our image.

1. **Implement Seccomp and AppArmor:**

We can apply Seccomp and AppArmor which is used to restrict the actions that containers can perform hence it would secure our cluster. We can directly apply both in a pod’s specification and restrict the restricted syscalls.

Seccomp is known as secure computing mode which helps to limit the system calls that a process can make. Syscalls are used to request some services from the operating system and Seccomp can restrict this or we can allow the set of system calls.

AppArmor is known as a mandatory access control (MAC) system which is used to restrict a process to perform any action. We create some policies which define the permissions given to a process and the files and directories it can access. This will ensure that the process is not performing any malicious activity.Let’s understand how we can implement it.

* We should load the necessary kernel modules on the nodes and then we can enable Seccomp and AppArmor in the cluster. Thes can also be applied on a container runtime such as Docker , by using –security-opt flag and mention the Seccomp/AppArmor profiles.
* Create Seccomp and AppArmor profiles by defining the permissions and set of system calls that a container can use. Seccomp profiles can be created in JSON format using tools like seccomp-tools whereas we can use aa-genprof or aa-logprof tools to create AppArmor profiles.
* Once our profiles are ready, we can apply them to pods and deployments by adding annotations referring to the Seccomp and AppArmor profiles.
* Once the profiles are applied to the pods, test them by performing the actions which are not permitted in Seccomp and AppArmor profiles.

1. **Monitoring and Auditing Kubernetes cluster**

It is important to monitor and audit the cluster’s activity to ensure that there is no security threat or everything is working properly. With monitoring, we can identify the issues and resolve the issue before it gets escalated. We can monitor the issues related to pod’s failure, resource constraints or latency. Whereas auditing helps to maintain security and compliance within a Kubernetes cluster. We can easily identify who did what on the cluster and ensure that all the actions are aligned with organizations policies. We can use various steps to do it which is explained below:

* We can use Kubernetes Dashboard which is WebUI which provides the visibility and control over the cluster. It gives a real-time view of the cluster’s state and detailed information about the resources. This dashboard can be used to monitor the cluster’s activity like status of the pods and nodes.
* Kubernetes can use monitoring tools like Prometheus and Grafana , which are used to gather and visualize the metrics. Further these metrics can be used to monitor the performance of our applications or servers and based on that we can detect anomalies and troubleshoot the issues. Whereas auditing collects the security relevant and set of records which happened sequentially within the cluster. We can create an audit policy which defines what events should be recorded and what information they should include.
* We can access container logs and identify the issues related to the application. We have tools like Kibana, Fluentd or Elasticsearch which can be used to analyze logs from our cluster.
* We should implement alerting to get alerts for critical issues. Tools like Prometheus, Alertmanager or Grafana Alerting can be used to set up the alert. We can set up alerts like high CPU utilization, nodes failure, pods failures etc. Alerts can further be integrated with receivers like email, messaging apps so that we can get the alert as soon as possible.
* We can use a tool called Falco to detect the threats and give us the notifications to work on. There are many applications which are continuously sending the syscall and it becomes difficult to monitor. Falco will help us monitor some of the critical syscalls

1. **Container Runtime Security**

Kubernetes supports a variety of container runtimes like Docker, containerd etc. These runtimes are responsible for managing and executing containers in a cluster. There are various methods which can be used to secure the containers.

* A container runtime class is an object in Kubernetes that instructs which runtime should be used for a pod. This is how we can implement container sandboxing which isolates containers from the operating system. We can use the gVisor tool from Google which creates another layer between the containers and kernel and all the syscalls go via gVisor. Whereas we have another approach which is called kata container which takes a different approach than gVisor. Kata installs each container in its own separate virtual machine and it needs support from virtualization.
* We can apply special security settings which are called Pod Security Context. It allows us to set the security context like user and group ID, file system permissions, Linux capabilities etc. For example, We can disable the privilege escalation and keep the readonly filesystem to true.

1. **Secure ETCD and secret data**

ETCD is a distributed key-value store which stores configuration data, service discovery information and other metadata which is necessary for our cluster to function properly. ETCD stores sensitive information such as secrets, TLS certificates, service account tokens which are critical for our cluster and if there is a security breach then it might lead to losing that sensitive information or service disruptions.

The secret information stored in ETCD database is base64 encoded, therefore not secured, since anyone having the permission to access secrets can easily decode the contents and see the sensitive information in plain text. We can use below methods to secure our ETCD database.

* We can implement TLS encryption between ETCD nodes and clients which will protect the data in transit.
* We have to implement AUthentication and Authorization so that only authorized users have the access to ETCD and secret data.
* We can also implement RBAC by providing granular permissions to a role and further that role should be linked to a user or a group.
* We can also use a key management system or a secure database to store our data and to prevent it from unauthorized access.
* We can also consider enabling encryption at rest to secure our sensitive information.

1. **Update and Patch Kubernetes components:**

Kubernetes is evolving rapidly to match the organization's requirements. So you would come across vulnerabilities and bugs on a regular basis. It is important to keep all Kubernetes components up to date to ensure the security of the cluster.

We have various control plane components which need to be updated to keep our cluster secure. We have tools like Kubeadm, Kubernetes Operations or Red Hat Openshift which can be used to apply the updates. But we need to test the updates in a non-production environment before applying to a production environment.

In conclusion, securing a containerized environment is a complex and ongoing process. Organizations would keep getting new security challenges, so we should apply all the best practices as per organization’s security policies. So it is important to review and update security policies on a regular basis. Auditing and monitoring should be implemented which would keep us informed of the latest security threats and vulnerabilities so that we can work on it proactively to mitigate any security concerns.