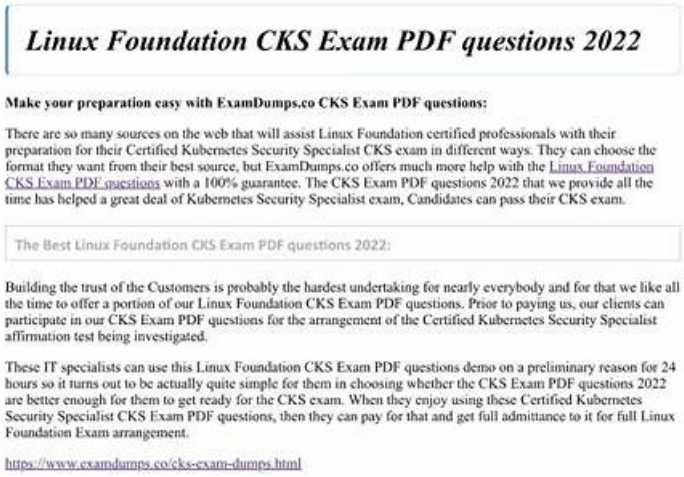
Free PDF Quiz Linux Foundation - CKS Newest New Dumps Free



Linux Foundation CKS Exam PDF questions 2022

Make your preparation easy with ExamDumps.co CKS Exam PDF questions:

There are so many sources on the web that will assist Linux Foundation certified professionals with their preparation for their Certified Kubernetes Security Specialist CKS exam in different ways. They can choose the format they want from their best source, but ExamDumps.co offers much more help with the Linux Foundation CKS Exam PDF questions with a 100% guarantee. The CKS Exam PDF questions 2022 that we provide all the time has helped a great deal of Kubernetes Security Specialist exam, Candidates can pass their CKS exam.

The Best Linux Foundation CKS Exam PDF questions 2022:

Building the trust of the Customers is probably the hardest undertaking for nearly everybody and for that we like all the time to offer a portion of our Linux Foundation CKS Exam PDF questions. Prior to paying us, our clients can participate in our CKS Exam PDF questions for the arrangement of the Certified Kubernetes Security Specialist affirmation test being investigated.

These IT specialists can use this Linux Foundation CKS Exam PDF questions demo on a preliminary reason for 24 hours so it turns out to be actually quite simple for them in choosing whether the CKS Exam PDF questions 2022 are better enough for them to get ready for the CKS exam. When they enjoy using these Certified Kubernetes Security Specialist CKS Exam PDF questions, then they can pay for that and get full admittance to it for full Linux Foundation Exam arrangement.

https://www.examdumps.co/cks-exam-dumps.html

DOWNLOAD the newest TestBraindump CKS PDF dumps from Cloud Storage for free: https://drive.google.com/open?id=1aCOU0GXLacuIR-xlnFAefK5sXcAkXMvA

Do you want to obtain your CKS study materials as quickly as possible? If you do, then we will be your best choice. You can receive downloading link and password with ten minutes after buying. In addition, CKS exam dumps are high quality, because we have experienced experts to edit, and you can pass your exam by using CKS Exam Materials of us. In addition, we are pass guarantee and money back guarantee, if you fail to pass the exam by using CKS study materials of us, we will give you full refund. And the money will be returned to your payment account.

Linux Foundation CKS (Certified Kubernetes Security Specialist) Exam is a certification exam that is designed to test the expertise of IT professionals in securing Kubernetes clusters. Kubernetes is a popular container orchestration tool that is used to manage and automate the deployment, scaling, and management of containerized applications. As Kubernetes becomes more widely adopted, the need for skilled IT professionals who can secure Kubernetes clusters has become increasingly important.

The CKS Certification Exam is a must-have credential for security specialists who are responsible for securing Kubernetes-based systems. Certified Kubernetes Security Specialist (CKS) certification demonstrates mastery of best security practices within Kubernetes environments, which is a critical competency for businesses that use cloud-native technologies. Passing the exam requires significant skill and hard work, but once obtained, this certification greatly increases job prospects and earning potential.

>> New CKS Dumps Free <<

Useful CKS – 100% Free New Dumps Free | CKS Pass Guide

The reality is often cruel. What do we take to compete with other people? More useful certifications like CKS certificate? In this era of surging talent, why should we stand out among the tens of thousands of graduates and be hired by the company? Perhaps the few qualifications you have on your hands are your greatest asset, and the CKS Test Prep is to give you that capital by passing exam fast and obtain certification soon. Don't doubt about it. More useful certifications mean more ways out. If you pass the CKS exam, you will be welcome by all companies which have relating business with CKS exam torrent.

The CKS certification exam is a performance-based exam that assesses the candidate's ability to perform tasks related to securing a Kubernetes cluster. CKS exam covers a wide range of topics, including cluster hardening, network security, identity and access management, and container security. CKS Exam is conducted online and is proctored, ensuring that the candidate's skills are evaluated fairly and accurately.

Linux Foundation Certified Kubernetes Security Specialist (CKS) Sample Questions (Q111-Q116):

NEW QUESTION # 111
Fix all issues via configuration and restart the affected components to ensure the new setting takes effect.
Fix all of the following violations that were found against the API server:- a. Ensure the --authorization-mode argument includes RBAC b. Ensure the --authorization-mode argument includes Node c. Ensure that the --profiling argument is set to false Fix all of the following violations that were found against the Kubelet:- a. Ensure the --anonymous-auth argument is set to false.
b. Ensure that the --authorization-mode argument is set to Webhook.
Fix all of the following violations that were found against the ETCD:-
a. Ensure that the --auto-tls argument is not set to true
Hint: Take the use of Tool Kube-Bench

Answer:

Explanation:
API server:
Ensure the --authorization-mode argument includes RBAC
Turn on Role Based Access Control. Role Based Access Control (RBAC) allows fine-grained control over the operations that different entities can perform on different objects in the cluster. It is recommended to use the RBAC authorization mode.
Fix - Buildtime
Kubernetes
apiVersion: v1
kind: Pod
metadata:
creationTimestamp: null
labels:
component: kube-apiserver
tier: control-plane
name: kube-apiserver
namespace: kube-system
spec:
containers:
- command:
+ - kube-apiserver
+ - --authorization-mode=RBAC,Node
image: gcr.io/google_containers/kube-apiserver-amd64:v1.6.0
livenessProbe:
failureThreshold: 8
httpGet:
host: 127.0.0.1
path: /healthz
port: 6443
scheme: HTTPS
initialDelaySeconds: 15
timeoutSeconds: 15
name: kube-apiserver-should-pass
resources:
requests:
cpu: 250m

volumeMounts:
- mountPath: /etc/kubernetes/
name: k8s
readOnly: true
- mountPath: /etc/ssl/certs
name: certs
- mountPath: /etc/pki
name: pki
hostNetwork: true
volumes:
- hostPath:
path: /etc/kubernetes
name: k8s
- hostPath:
path: /etc/ssl/certs
name: certs
- hostPath:
path: /etc/pki
name: pki
Ensure the --authorization-mode argument includes Node
Remediation: Edit the API server pod specification file /etc/kubernetes/manifests/kube-apiserver.yaml on the master node and set the --authorization-mode parameter to a value that includes Node.
--authorization-mode=Node,RBAC
Audit:
/bin/ps -ef | grep kube-apiserver | grep -v grep
Expected result:
'Node,RBAC' has 'Node'
Ensure that the --profiling argument is set to false
Remediation: Edit the API server pod specification file /etc/kubernetes/manifests/kube-apiserver.yaml on the master node and set the below parameter.
--profiling=false
Audit:
/bin/ps -ef | grep kube-apiserver | grep -v grep
Expected result:
'false' is equal to 'false'
Fix all of the following violations that were found against the Kubelet:- Ensure the --anonymous-auth argument is set to false.
Remediation: If using a Kubelet config file, edit the file to set authentication: anonymous: enabled to false. If using executable arguments, edit the kubelet service file /etc/systemd/system/kubelet.service.d/10-kubeadm.conf on each worker node and set the below parameter in KUBELET_SYSTEM_PODS_ARGS variable.
--anonymous-auth=false
Based on your system, restart the kubelet service. For example:
systemctl daemon-reload
systemctl restart kubelet.service
Audit:
/bin/ps -fC kubelet
Audit Config:
/bin/cat /var/lib/kubelet/config.yaml
Expected result:
'false' is equal to 'false'
2) Ensure that the --authorization-mode argument is set to Webhook.
Audit
docker inspect kubelet | jq -e '.[0].Args[] | match("--authorization-mode=Webhook").string' Returned Value: --authorization-mode=Webhook Fix all of the following violations that were found against the ETCD:- a. Ensure that the --auto-tls argument is not set to true Do not use self-signed certificates for TLS. etcd is a highly-available key value store used by Kubernetes deployments for persistent storage of all of its REST API objects. These objects are sensitive in nature and should not be available to unauthenticated clients. You should enable the client authentication via valid certificates to secure the access to the etcd service.
Fix - Buildtime
Kubernetes
apiVersion: v1
kind: Pod
metadata:

annotations:
scheduler.alpha.kubernetes.io/critical-pod: ""
creationTimestamp: null
labels:
component: etcd
tier: control-plane
name: etcd
namespace: kube-system
spec:
containers:
- command:
+ - etcd
+ - --auto-tls=true
image: k8s.gcr.io/etcd-amd64:3.2.18
imagePullPolicy: IfNotPresent
livenessProbe:
exec:
command:
- /bin/sh
- -ec
- ETCDCTL_API=3 etcdctl --endpoints=https://[192.168.22.9]:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt
--cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt --key=/etc/kubernetes/pki/etcd/healthcheck-client.key get foo
failureThreshold: 8 initialDelaySeconds: 15 timeoutSeconds: 15 name: etcd-should-fail resources: {} volumeMounts:
- mountPath: /var/lib/etcd
name: etcd-data
- mountPath: /etc/kubernetes/pki/etcd
name: etcd-certs
hostNetwork: true
priorityClassName: system-cluster-critical
volumes:
- hostPath:
path: /var/lib/etcd
type: DirectoryOrCreate
name: etcd-data
- hostPath:
path: /etc/kubernetes/pki/etcd
type: DirectoryOrCreate
name: etcd-certs
status: {}

NEW QUESTION # 112
You need to implement a secure CI/CD pipeline for building and deploying containerized applications to a Kubernetes cluster. The pipeline should include security checks and validation steps at each stage to minimize the risk of introducing vulnerabilities. What security best practices would you follow?

Answer:

Explanation:
Solution (Step by Step) :
1. Source Code Security:
- Static Application Security Testing (SAST): Integrate SAST tools into your CIICD pipeline to identify vulnerabilities in your source code.
- Dependency Scanning: Use dependency scanning tools to identify known vulnerabilities in your application's dependencies.
- Code Review: Enforce mandatory code reviews for all changes to production branches to catch potential vulnerabilities.
2. Container Image Security'
- Container Image Scanning: Scan your container images for vulnerabilities and malware.
- Multi-stage Builds: Use multi-stage Docker builds to create smaller and more secure container images.
- Signed Images: Sign your container images to ensure their authenticity and prevent tampering.
3. Infrastructure Security:
- Infrastructure as Code (IaC): Use Iac tools to define your Kubernetes infrastructure and configurations, ensuring consistency and

security.
- Policy Enforcement: Implement Kubernetes admission controllers and policies to enforce security best practices during deployment.
4. Deployment Security:
- Role-Based Access Control (RBAC): Use RBAC to restrict access to sensitive Kubernetes resources.
- Network Policies: Implement network policies to control communication between pods.
- Deployment Strategies: Choose deployment strategies like rolling updates or canary deployments to minimize the impact of security incidents.
5. Monitoring and Auditing:
- Kubernetes Logging and Monitoring: Configure logging and monitoring to track events and identify potential security incidents.
- Security Auditing: Regularly audit your CI/CD pipeline and Kubernetes cluster for security compliance.
6. Continuous Security Assessment:
- Security Scanning: Regularly scan your source code, container images, and infrastructure for vulnerabilities.
- Vulnerability Management Track and remediate discovered vulnerabilities.
7. Secure Development Practices:
- Secure Coding Standards: Enforce secure coding standards and best practices.
- Security Training: Provide security training to developers to increase awareness of common vulnerabilities.
- Security Bug Bounties: Consider offering security bug bounties to incentivize ethical hackers to find and report vulnerabilities.

```
# Example GitLab CI/CD YAML file for secure containerized deployment
stages:
  - build
  - test
  - scan
  - deploy

variables:
  # Configure your security tools
  # Example: Snyk for dependency scanning and container image scanning
  SNYK_TOKEN: "$SNYK_TOKEN"

build:
  stage: build
  image: docker:latest
  script:
    - docker build -t my-app .
    - docker push my-app

test:
  stage: test
  image: node:latest
  script:
    - npm install
    - npm test

scan:
  stage: scan
  image: snyk/snyk
  script:
    - snyk test
    - snyk container test my-app:latest
  # Configure snyk to fail the pipeline if vulnerabilities are found
  environment:
    SNYK_TOKEN: $SNYK_TOKEN

deploy:
  stage: deploy
  image: docker:latest
  script:
    - docker login -u $DOCKER_USER -p $DOCKER_PASSWORD $DOCKER_REGISTRY
    - docker push my-app:latest
    - kubectl apply -f deployment.yaml
```


NEW QUESTION # 113

SIMULATION
use the Trivy to scan the following images,
1. amazonlinux:1
2. k8s.gcr.io/kube-controller-manager:v1.18.6
Look for images with HIGH or CRITICAL severity vulnerabilities and store the output of the same in /opt/trivy-vulnerable.txt

- A. Send us the Feedback on it.

Answer: A

NEW QUESTION # 114
Service is running on port 389 inside the system, find the process-id of the process, and stores the names of all the open-files inside the /candidate/KH77539/files.txt, and also delete the binary.

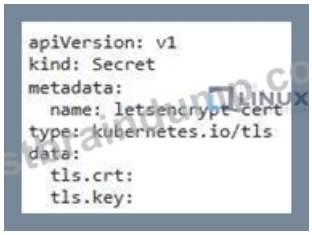
Answer:

Explanation:
root# netstat -ltnup
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name tcp 0 0 127.0.0.1:17600 0.0.0.0:* LISTEN 1293/dropbox tcp 0 0 127.0.0.1:17603 0.0.0.0:* LISTEN 1293/dropbox tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 575/sshd tcp 0 0 127.0.0.1:9393 0.0.0.0:* LISTEN 900/perl tcp 0 0 :::80 :::* LISTEN 9583/docker-proxy tcp 0 0 :::443 :::* LISTEN 9571/docker-proxy udp 0 0 0.0.0.0:68 0.0.0.0:* 8822/dhcpcd
...
root# netstat -ltnup | grep ':22'
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 575/sshd
The ss command is the replacement of the netstat command.
Now let's see how to use the ss command to see which process is listening on port 22:
root# ss -ltnup 'sport = :22'
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port
tcp LISTEN 0 128 0.0.0.0:22 0.0.0.0:* users:("sshd",pid=575,fd=3))

NEW QUESTION # 115
You have a Kubernetes cluster with a deployment named 'myapp' that serves a web application. You want to secure the application by implementing HTTPS using Ingress and TLS certificates. You have Obtained a TLS certificate from Let's Encrypt and stored it in a Kubernetes secret named 'letsencrypt-cert'. Configure an Ingress resource to expose the application on the domain 'myapp.example.com' with HTTPS enabled, using the 'letsencrypt-celt' secret.

Answer:

Explanation:
Solution (Step by Step) :
1. Create a Kubernetes secret for the Let's Encrypt certificate:



- Replace and with the actual certificate and private key data, respectively. 2. Create an Ingress resource:

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: myapp-ingress
spec:
  tls:
  - hosts:
    - myapp.example.com
    secretName: letsencrypt-cert
  rules:
  - host: myapp.example.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: myapp-service
            port:
              number: 80
```

- Replace 'myapp-service' and '80' with the actual service name and pod number of your application. 3. Apply the Ingress and Secret resources: bash kubectl apply -f letsencrypt-cert.yaml kubectl apply -f myapp-ingress.yaml 4. Verify the Ingress configuration: bash kubectl get ingress myapp-ingress - Check that the Ingress resource has been created and the status is "Ready". 5. Access the application through HTTPS: - You can now access the web application securely through HTTPS using the domain 'myapp.example.com' Note: - Ensure that the 'myapp-service' is created and running before applying the Ingress resource. - Make sure that your cluster has the 'Ingress' controller installed. - If you are using a different Certificate Authority, modify the 'secretName' in the Ingress resource accordingly.

NEW QUESTION # 116

......