CKS Q&As

Certified Kubernetes Security Specialist (CKS) Exam

Pass Linux Foundation CKS Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.pass4itsure.com/cks.html

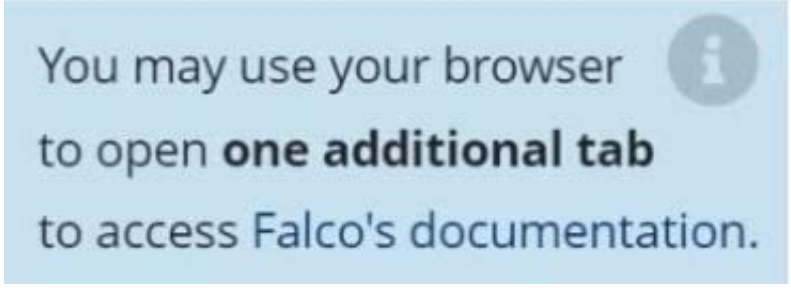
100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Linux Foundation Official Exam Center

QUESTION 1

CORRECT TEXT


You must complete this task on the following cluster/nodes:
Cluster
Master node
Worker node
KSRS00101
ksrs00101-master
ksrs00101-worker1
You can switch the cluster/configuration context using the following command:
[candidate@cli] $ kubectl config use-context KSRS00101


You may use your browser to open one additional tab to access Falco's documentation.

Two tools are pre-installed on the cluster\\'s worker node:

1.

sysdig

2.

falco

Using the tool of your choice (including any non pre-installed tool), analyze the container\\'s behavior for at least 30 seconds, using filters that detect newly spawning and executing processes. Store an incident file at /opt/KSRS00101/alerts/

details, containing the detected incidents, one per line, in the following format:

```
timestamp,uid/username,proce
ssName
```

The following example shows a properly formatted incident file:

```
01:40:19.601363716,root,init
01:40:20.606013716,nobody,ba
sh
01:40:21.137163716,1000,tar
```

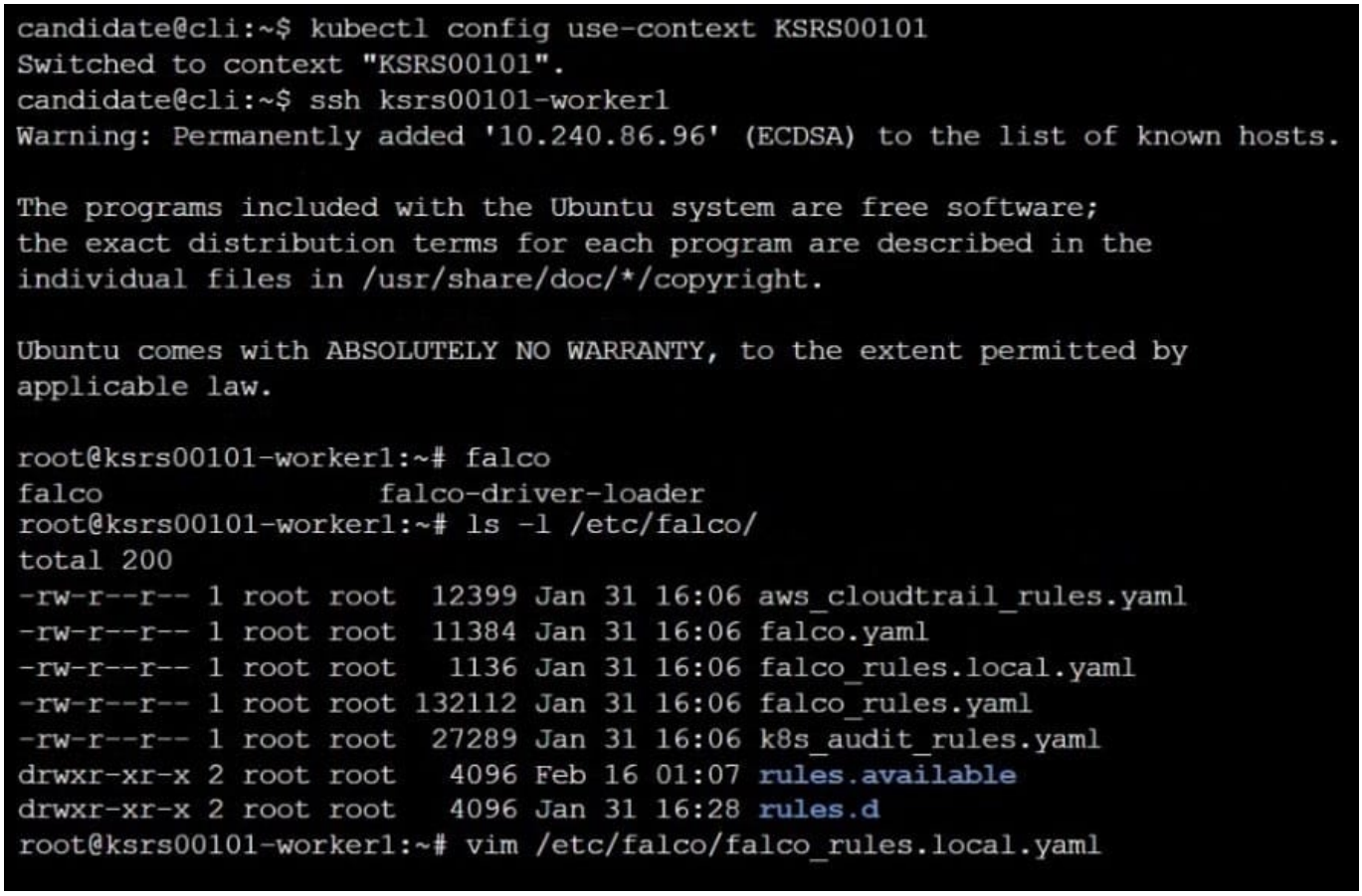
Keep the tool's original timestamp-format as-is.

Make sure to store the incident file on the cluster's worker node.

A. See the explanation below:

B. PlaceHolder

Correct Answer: A



```
# Copyright (C) 2019 The Falco Authors.
#
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

####################
# Your custom rules!
####################

# Add new rules, like this one
# - rule: The program "sudo" is run in a container
#   desc: An event will trigger every time you run sudo in a container
#   condition: evt.type = execve and evt.dir=< and container.id != host and proc.name = sudo
#   output: "Sudo run in container (user=%user.name %container.info parent=%proc.pname cmdli
ne=%proc.cmdline)"
#   priority: ERROR
#   tags: [users, container]

# Or override/append to any rule, macro, or list from the Default Rules
- rule: Container Drift Detected (chmod)
  desc: New executable created in a container due to chmod
  condition: >
    evt.type in (open,openat,create) and
    evt.is_open_exec=true and
    container and
    not runc_writing_exec_fifo and
    not runc_writing_var_lib_docker and
    not user_known_container_drift_activities and
    evt.rawres>=0
  output:
    %evt.time,%user.uid,%proc.name
  priority: ERROR
```

Text

```
root@ksrs00101-worker1:~# vim /etc/falco/falco_rules.local.yaml
root@ksrs00101-worker1:~# systemctl status falco.service
● falco.service - Falco Runtime Security
     Loaded: loaded (/lib/systemd/system/falco.service; disabled; vendor preset: enabled)
     Active: inactive (dead)
root@ksrs00101-worker1:~# systemctl enable falco.service
Created symlink /etc/systemd/system/multi-user.target.wants/falco.service → /lib/systemd/sys
tem/falco.service.
root@ksrs00101-worker1:~# systemctl start falco.service
root@ksrs00101-worker1:~# exit
logout
Connection to 10.240.86.96 closed.
candidate@cli:~$ ssh ksrs00101-worker1
Last login: Fri May 20 15:59:48 2022 from 10.240.86.88
root@ksrs00101-worker1:~# vim /etc/falco/falco.yaml
```

```
# When using json output, whether or not to include the "tags" property
# itself in the json output. If set to true, outputs caused by rules
# with no tags will have a "tags" field set to an empty array. If set to
# false, the "tags" field will not be included in the json output at all.
json_include_tags_property: true

# Send information logs to stderr and/or syslog Note these are *not* security
# notification logs! These are just Falco lifecycle (and possibly error) logs.
log_stderr: true
log_syslog: true
log_file: /opt/KSRS00101/alerts/details

# Minimum log level to include in logs. Note: these levels are
# separate from the priority field of rules. This refers only to the
# log level of falco's internal logging. Can be one of "emergency",
# "alert", "critical", "error", "warning", "notice", "info", "debug".
log_level: info
```

Text

```
root@ksrs00101-worker1:~# vim /etc/falco/falco.yaml
root@ksrs00101-worker1:~# grep log /etc/falco/falco.yaml
# cloudtrail log files.
# If true, the times displayed in log messages and output messages
# Send information logs to stderr and/or syslog Note these are *not* security
# notification logs! These are just Falco lifecycle (and possibly error) logs.
log_stderr: true
log_syslog: true
log_file: /opt/KSRS00101/alerts/details
# Minimum log level to include in logs. Note: these levels are
# log level of falco's internal logging. Can be one of "emergency",
log_level: info
#    - log: log a DEBUG message noting that the buffer was full
# Notice it is not possible to ignore and log/alert messages at the same time.
# The rate at which log/alert messages are emitted is governed by a
    - log
# The timeout error will be reported to the log according to the above log_* settings.
syslog_output:
#    - logging (alternate method than syslog):
#         program: logger -t falco-test
# this information will be logged, however the main Falco daemon will not be stopped.
root@ksrs00101-worker1:~# systemctl restart falco.service
root@ksrs00101-worker1:~# exit
logout
Connection to 10.240.86.96 closed.
candidate@cli:~$
```

QUESTION 2

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: pod-access
  namespace: dev-team
spec:
  podSelector:
    matchLabels:
      environment: dev
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              environment: dev
        - podSelector:
            matchLabels:
              environment: testing
```

```
candidate@cli:~$ vim np.yaml
candidate@cli:~$ cat np.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: pod-access
  namespace: dev-team
spec:
  podSelector:
    matchLabels:
      environment: dev
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              environment: dev
        - podSelector:
            matchLabels:
              environment: testing
candidate@cli:~$
candidate@cli:~$
candidate@cli:~$ kubectl create -f np.yaml -n dev-team
networkpolicy.networking.k8s.io/pod-access created
candidate@cli:~$ kubectl describe netpol -n dev-team
Name:         pod-access
Namespace:    dev-team
Created on:   2022-05-20 15:35:33 +0000 UTC
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:     environment=dev
  Allowing ingress traffic:
    To Port: <any> (traffic allowed to all ports)
    From:
      NamespaceSelector: environment=dev
    From:
      PodSelector: environment=testing
  Not affecting egress traffic
  Policy Types: Ingress
candidate@cli:~$ cat KSSH00301/network-policy.yaml
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ""
  namespace: ""
spec:
  podSelector: {}
  policyTypes:
    - Ingress
  ingress:
    - from: []
    - from: []
candidate@cli:~$ cp np.yaml KSSH00301/network-policy.yaml
candidate@cli:~$ cat KSSH00301/network-policy.yaml
```

```
candidate@cli:~$ cat KSSH00301/network-policy.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: pod-access
  namespace: dev-team
spec:
  podSelector:
    matchLabels:
      environment: dev
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              environment: dev
        - podSelector:
            matchLabels:
              environment: testing
candidate@cli:~$
```

Enable audit logs in the cluster, To Do so, enable the log backend, and ensure that

1.

logs are stored at /var/log/kubernetes-logs.txt.

2.

Log files are retained for 12 days.

3.

at maximum, a number of 8 old audit logs files are retained.

4.

set the maximum size before getting rotated to 200MB

Edit and extend the basic policy to log:

1.

namespaces changes at RequestResponse

2.

Log the request body of secrets changes in the namespace kube-system.

3.

Log all other resources in core and extensions at the Request level.

4.

Log "pods/portforward", "services/proxy" at Metadata level.

5.

Omit the Stage RequestReceived

All other requests at the Metadata level

A. See the explanation below:

B. PlaceHolder

Correct Answer: A

Kubernetes auditing provides a security-relevant chronological set of records about a cluster. Kube-apiserver performs auditing. Each request on each stage of its execution generates an event, which is then pre-processed according to a

certain policy and written to a backend. The policy determines what\\'s recorded and the backends persist the records. You might want to configure the audit log as part of compliance with the CIS (Center for Internet Security) Kubernetes

Benchmark controls.

The audit log can be enabled by default using the following configuration in cluster.yml:

services:

kube-api:

audit_log:

enabled: true

When the audit log is enabled, you should be able to see the default values at /etc/kubernetes/audit-policy.yaml

The log backend writes audit events to a file in JSONlines format. You can configure the log audit backend using the following kube-apiserver flags:

--audit-log-path specifies the log file path that log backend uses to write audit events. Not specifying this flag disables log backend. - means standard out --audit-log-maxage defined the maximum number of days to retain old audit log files

--audit-log-maxbackup defines the maximum number of audit log files to retain

--audit-log-maxsize defines the maximum size in megabytes of the audit log file before it gets rotated

If your cluster\\'s control plane runs the kube-apiserver as a Pod, remember to mount the hostPath to the location of the policy file and log file, so that audit records are persisted.

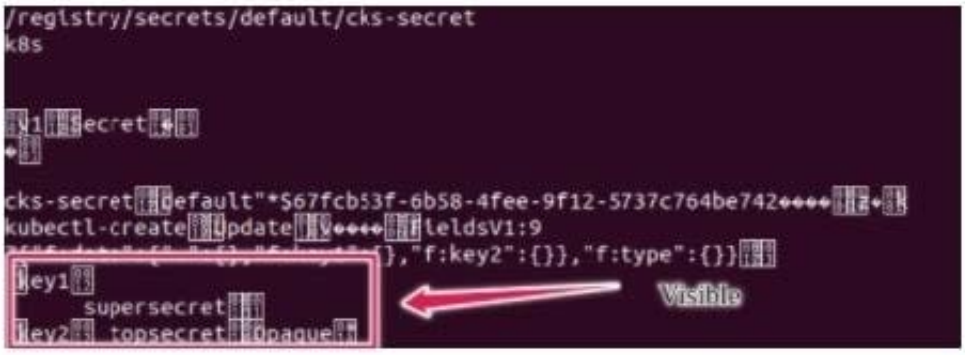
For example:

--audit-policy-file=/etc/kubernetes/audit-policy.yaml \

--audit-log-path=/var/log/audit.log

QUESTION 3

Secrets stored in the etcd is not secure at rest, you can use the etcdctl command utility to find the secret value for e.g:ETCDCTL_API=3 etcdctl get /registry/secrets/default/cks-secret --cacert="ca.crt" -- cert="server.crt" --key="server.key" Output



Using the Encryption Configuration, Create the manifest, which secures the resource secrets using the provider AES-CBC and identity, to encrypt the secret-data at rest and ensure all secrets are encrypted with the new configuration.

A. See explanation below.

B. PlaceHolder

Correct Answer: A

1.

ETCD secret encryption can be verified with the help of etcdctl command line utility.

2.

ETCD secrets are stored at the path /registry/secrets/$namespace/$secret on the master node.

3.

The below command can be used to verify if the particular ETCD secret is encrypted or not.

ETCDCTL_API=3 etcdctl get /registry/secrets/default/secret1 [...] | hexdump -C

QUESTION 4

You can switch the cluster/configuration context using the following command:

[desk@cli] $ kubectl config use-context prod-account

Context:

A Role bound to a Pod\\'s ServiceAccount grants overly permissive permissions. Complete the following tasks to reduce the set of permissions.

Task:

Given an existing Pod named web-pod running in the namespace database.

1.

Edit the existing Role bound to the Pod\\'s ServiceAccount test-sa to only allow performing get operations, only on resources of type Pods.

2.

Create a new Role named test-role-2 in the namespace database, which only allows performing update operations, only on resources of type statuefulsets.

3.

Create a new RoleBinding named test-role-2-bind binding the newly created Role to the Pod\\'s ServiceAccount. Note: Don\\'t delete the existing RoleBinding.

A. See the explanation below

B. PlaceHolder

Correct Answer: A

```
candidate@cli:~$ kubectl config use-context KSCH00201
Switched to context "KSCH00201".
candidate@cli:~$ kubectl get pods -n security
NAME      READY   STATUS    RESTARTS   AGE
web-pod   1/1     Running   0          6h9m
candidate@cli:~$ kubectl get deployments.apps -n security
No resources found in security namespace.
candidate@cli:~$ kubectl describe rolebindings.rbac.authorization.k8s.io -n security
Name:         dev-role
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  dev-role
Subjects:
  Kind            Name      Namespace
  ----            ----      ---------
  ServiceAccount  sa-dev-1
candidate@cli:~$ kubectl describe role dev-role -n security
Name:         dev-role
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  *          []                 []              [*]
candidate@cli:~$ kubectl edit role/dev-role -n security
```

```
  uid: b4c9ddd6-2729-43bd-8fbd-b2d227f4c4cd
rules:
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - watch
```

```
candidate@cli:~$ kubectl describe role dev-role -n security
Name:         dev-role
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  *          []                 []              [*]
candidate@cli:~$ kubectl edit role/dev-role -n security
role.rbac.authorization.k8s.io/dev-role edited
candidate@cli:~$ kubectl describe role dev-role -n security
Name:         dev-role
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  services   []                 []              [watch]
candidate@cli:~$ kubectl get pods -n security
NAME      READY   STATUS    RESTARTS   AGE
web-pod   1/1     Running   0          6h12m
candidate@cli:~$ kubectl get pods/web-pod -n security -o yaml | grep serviceAccount
  serviceAccount: sa-dev-1
  serviceAccountName: sa-dev-1
      - serviceAccountToken:
candidate@cli:~$ kubectl create role role-2 --verb=update --resource=namespaces -n security
role.rbac.authorization.k8s.io/role-2 created
candidate@cli:~$ kubectl create rolebinding role-2-binding --role
--role    --role=
candidate@cli:~$ kubectl create rolebinding role-2-binding --role=role-2 --serviceaccount=se
curity:sa-dev-1 -n security
rolebinding.rbac.authorization.k8s.io/role-2-binding created
candidate@cli:~$
```

QUESTION 5

Create a network policy named allow-np, that allows pod in the namespace staging to connect to port 80 of other pods in the same namespace.

Ensure that Network Policy:

1.

Does not allow access to pod not listening on port 80.

2.

Does not allow access from Pods, not in namespace staging.

A. See the explanation below:

B. PlaceHolder

Correct Answer: A

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: network-policy

spec:

podSelector: {} #selects all the pods in the namespace deployed policyTypes:

-Ingress ingress:

-ports: #in input traffic allowed only through 80 port only

-protocol: TCP port: 80

QUESTION 6

Create a PSP that will prevent the creation of privileged pods in the namespace.

Create a new PodSecurityPolicy named prevent-privileged-policy which prevents the creation of privileged pods.

Create a new ServiceAccount named psp-sa in the namespace default.

Create a new ClusterRole named prevent-role, which uses the newly created Pod Security Policy prevent-privileged-policy.

Create a new ClusterRoleBinding named prevent-role-binding, which binds the created ClusterRole prevent-role to the created SA psp-sa.

Also, Check the Configuration is working or not by trying to Create a Privileged pod, it should get failed.

A. See the below.

B. PlaceHolder

Correct Answer: A

Create a PSP that will prevent the creation of privileged pods in the namespace. $ cat clusterrole-use-privileged.yaml apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole metadata: name: use-privileged-psp rules:

-apiGroups: [\\'policy\\']

resources: [\\'podsecuritypolicies\\']

verbs: [\\'use\\']

resourceNames:

-default-psp

apiVersion: rbac.authorization.k8s.io/v1 kind: RoleBinding metadata: name: privileged-role-bind namespace: psp-test roleRef: apiGroup: rbac.authorization.k8s.io kind: ClusterRole name: use-privileged-psp subjects:

-kind: ServiceAccount name: privileged-sa $ kubectl -n psp-test apply -f clusterrole-use-privileged.yaml

After a few moments, the privileged Pod should be created.

Create a new PodSecurityPolicy named prevent-privileged-policy which prevents the creation of privileged pods.

apiVersion: policy/v1beta1

kind: PodSecurityPolicy

metadata:

name: example

spec:

privileged: false # Don\\'t allow privileged pods!

The rest fills in some required fields.

seLinux:

rule: RunAsAny

supplementalGroups:

rule: RunAsAny

runAsUser:

rule: RunAsAny

fsGroup:

rule: RunAsAny

volumes:

-\\'*\\'

And create it with kubectl:

kubectl-admin create -f example-psp.yaml

Now, as the unprivileged user, try to create a simple pod:

kubectl-user create -f-