SECRET-SEN Q&As

CyberArk Sentry - Secrets Manager

Pass CyberArk SECRET-SEN Exam with 100% Guarantee

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

https://www.geekcert.com/secret-sen.html

100% Passing Guarantee
100% Money Back Assurance

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

- **Instant Download** After Purchase
- **100% Money Back** Guarantee
- **365 Days** Free Update
- **800,000+** Satisfied Customers

QUESTION 1

You are setting up a Kubernetes integration with Conjur. With performance as the key deciding factor, namespace and service account will be used as identity characteristics.

Which authentication method should you choose?

A. JWT-based authentication

B. Certificate-based authentication

C. API key authentication

D. Connect (OIDC) authentication

Correct Answer: A

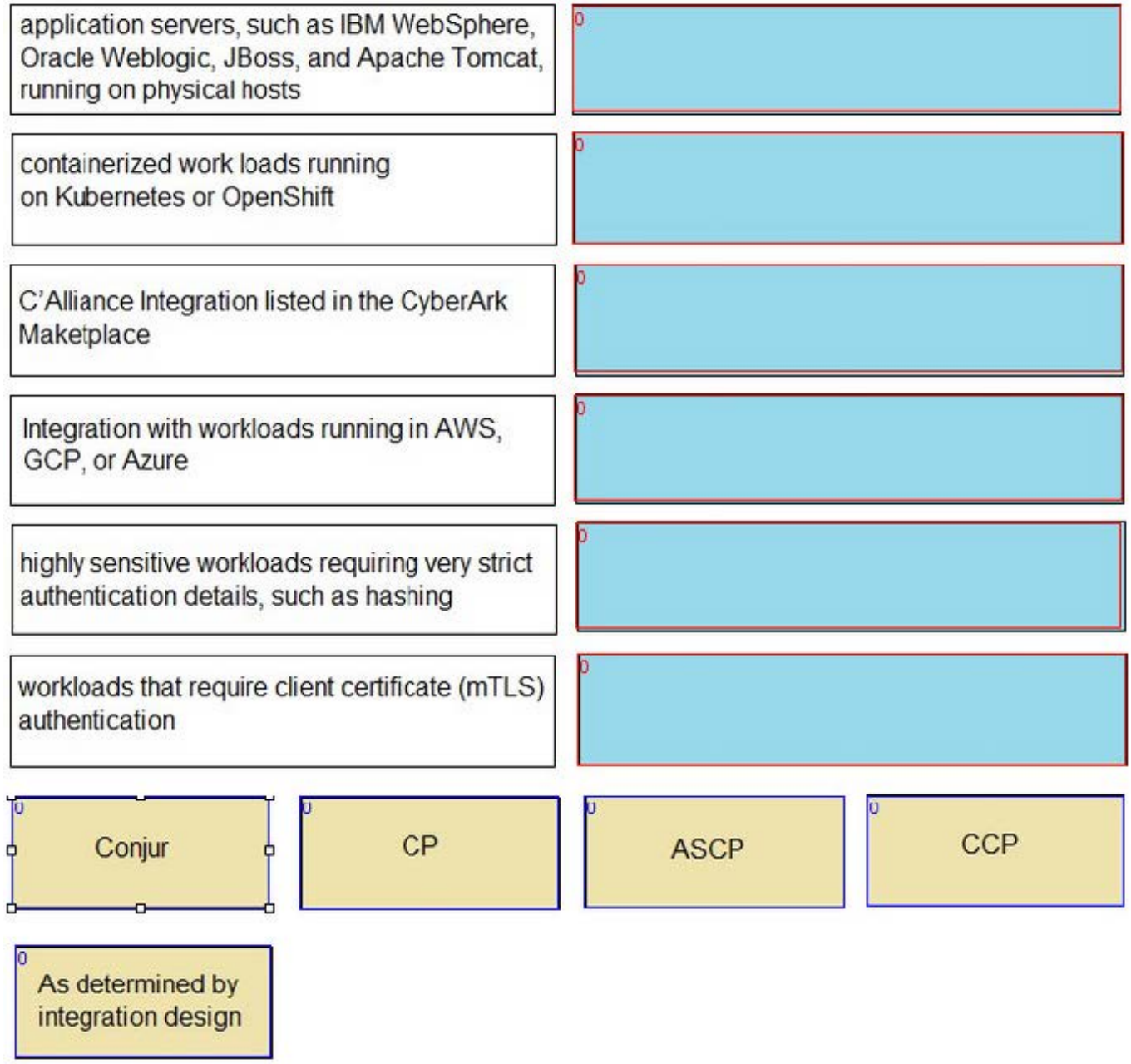
According to the CyberArk Sentry Secrets Manager documentation, JWT- based authentication is the recommended method for authenticating Kubernetes pods with Conjur. JWT-based authentication uses JSON Web Tokens (JWTs) that are issued by the Kubernetes API server and signed by its private key. The JWTs contain the pod\\'s namespace and service account as identity characteristics, which are verified by Conjur against a policy that defines the allowed namespaces and service accounts. JWT-based authentication is fast, scalable, and secure, as it does not require any additional certificates, secrets, or sidecars to be deployed on the pods. JWT-based authentication also supports rotation and revocation of the Kubernetes API server\\'s private key, which enhances the security and resilience of the authentication process. Certificate-based authentication is another method for authenticating Kubernetes pods with Conjur, but it is not the best option for performance. Certificate-based authentication uses X.509 certificates that are generated by a Conjur CA service and injected into the pods as Kubernetes secrets. The certificates contain the pod\\'s namespace and service account as identity characteristics, which are verified by Conjur against a policy that defines the allowed namespaces and service accounts. Certificate-based authentication is secure and reliable, but it requires more resources and steps to generate, inject, and manage the certificates and secrets. Certificate-based authentication also does not support rotation and revocation of the certificates, which may pose a security risk if the certificates are compromised or expired. API key authentication and Connect (OIDC) authentication are not valid methods for authenticating Kubernetes pods with Conjur. API key authentication is used for authenticating hosts, users, and applications that have a Conjur identity and an API key. Connect (OIDC) authentication is used for authenticating users and applications that have an OpenID Connect identity and a token. These methods are not suitable for Kubernetes pods, as they do not use the pod\\'s namespace and service account as identity characteristics, and they require additional secrets or tokens to be stored and managed on the pods. References: = JWT Authenticator | CyberArk Docs; Certificate Authenticator | CyberArk Docs; API Key Authenticator | CyberArk Docs; Connect Authenticator | CyberArk Docs

QUESTION 2

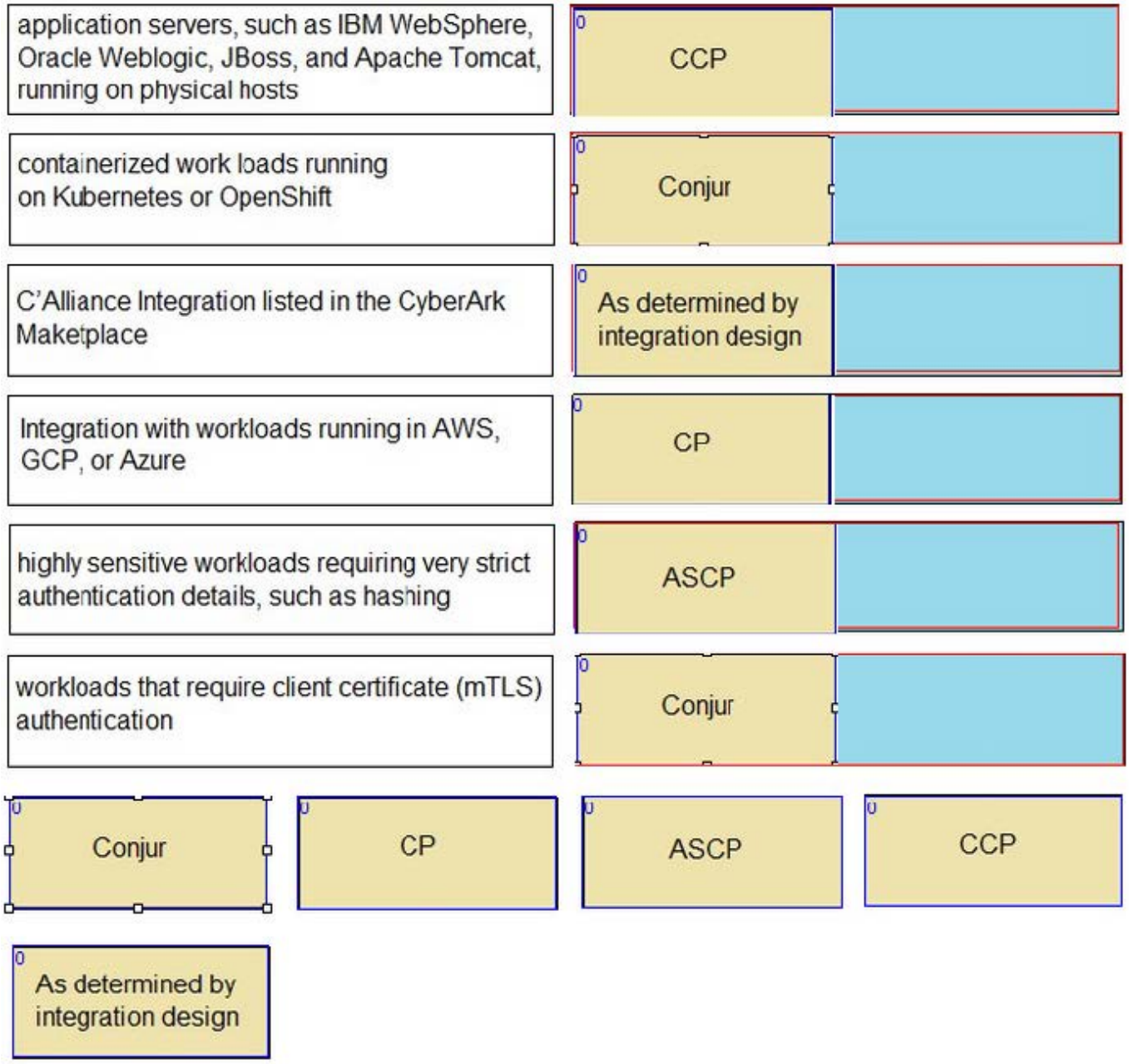
DRAG DROP

Match each use case to the appropriate Secrets Manager Solution.

Select and Place:



Correct Answer:


application servers, such as IBM WebSphere, Oracle Weblogic, JBoss, and Apache Tomcat, running on physical hosts
CCP
containerized work loads running on Kubernetes or OpenShift
Conjur
C'Alliance Integration listed in the CyberArk Maketplace
As determined by integration design
Integration with workloads running in AWS, GCP, or Azure
CP
highly sensitive workloads requiring very strict authentication details, such as hashing
ASCP
workloads that require client certificate (mTLS) authentication
Conjur
Conjur
CP
ASCP
CCP
As determined by integration design

Use Case:	Solution:
Application servers running on physical hosts	CCP
Containerized workloads running on Kubernetes/OpenShift	Conjur
C'Alliance Integration listed in the CyberArk Marketplace	As determined by integration design
Integration with workloads in AWS/GCP/Azure	CP
Highly sensitive workloads requiring strict authentication	ASCP
Workloads requiring client certificate (mTLS) authentication	Conjur or CCP

QUESTION 3

DRAG DROP

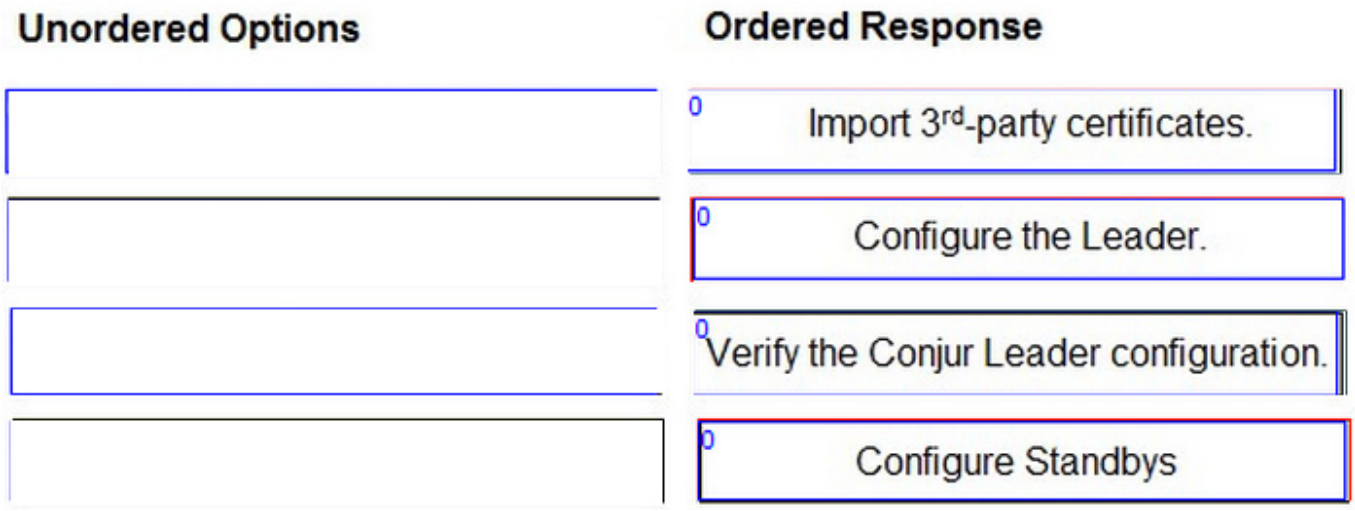
You are configuring the Conjur Cluster with 3rd-party certificates.

Arrange the steps to accomplish this in the correct sequence.

Select and Place:

Correct Answer:

Answer Area



The correct sequence of steps to configure the Conjur Cluster with 3rd-party certificates is as follows: Import 3rd-party certificates to the Leader using the command: docker exec mycontainer evoke ca import --force --root --chain 1 Configure the Leader using the command: docker exec mycontainer evoke configure master --accept-eula --hostname --admin-password 1 Verify the Conjur Leader configuration using the command: docker exec mycontainer evoke role Configure the Standbys using the command: docker exec mycontainer evoke configure standby --master-address --master-fingerprint 1 References: Certificate requirements

QUESTION 4

You have a PowerShell script that is being used on 1000 workstations. It requires a Windows Domain credential that is currently hard coded in the script.

What is the simplest solution to remove that credential from the Script?

A. Modify the script to use the CLI SDK to fetch the secret at runtime using Credential Providers installed on each workstation.

B. Modify the script to make a SOAP call to retrieve the secret from the Central Credential Provider.

C. Modify the script to run on WebSphere using the Application Server Credential Provider to retrieve the secret.

D. Use Conjur Summon to invoke the script and inject the secret at run time.

Correct Answer: D

Conjur Summon is an open source utility that can fetch secrets from Conjur and export them as environment variables to a sub-process environment. This way, the secrets are not exposed or stored in the script, but are only available at run time. To use Conjur Summon, you need to install the summon-conjur provider on each workstation, define the secrets in a secrets.yml file, and wrap the PowerShell script in summon. For example, if the secret ID is win/domain/cred, the

secrets.yml file would look like this:

DOMAIN_CRED: !var win/domain/cred

And the summon command would look like this:

summon --provider summon-conjur powershell script.ps1 This will inject the secret value of win/domain/cred as an environment variable named DOMAIN_CRED to the PowerShell script. The script can then access the secret using the

$env:DOMAIN_CRED syntax.

References: Summon-inject secrets, cyberark/summon-conjur

QUESTION 5

You are diagnosing this log entry: From Conjur logs:

```
USERNAME_MISSING failed to authenticate with authenticator authn-jwt service team-
a:webservice:conjur/
authn-jwt/jenkins-test: CONJ00087E Failed to fetch JWKS from
'https://jenkins.tst.acme.com/jwtauth/conjur
jwk-set'. Reason: '#<OpenSSL::SSL::SSLError:SSL_connect returned=1 errno=o state=error:
certificate verify
failed (unable to get local issuer certificate)>'

Apr 25, 2022 11:35:06 AM FINE org.conjur.jenkins.jwauth.impl.JwtToken sign
Signing Token
Apr 25, 2022 11:35:07 AM FINE org.conjur.jenkins.api.ConjurAPI getAuthorizationToken
Authenticating with Conjur (JWT) authnPath=authn-jwt/jenkins-test
Apr 25, 2022 11:35:08 AM FINEST org.conjur.jenkins.api.ConjurAPI getAuthorizationToken
Conjur Authenticate response 401 – Unauthorized
Apr 25, 2022 11:35:08 AM FINE org.conjur.jenkins.credentials.CredentialsSupplier get
EXCEPTION: CredentialSuplier => Error authenticating to Conjur [401 – Unauthorized
```

Given these errors, which problem is causing the breakdown?

A. The Jenkins certificate chain is not trusted by Conjur.

B. The Conjur certificate chain is not trusted by Jenkins.

C. The JWT sent by Jenkins does not match the Conjur host annotations.

D. The Jenkins certificate is malformed and will not be trusted by Conjur.

Correct Answer: A

The log entry shows a failed authentication attempt with Conjur using the authn-jwt method. This method allows applications to authenticate with Conjur using JSON Web Tokens (JWTs) that are signed by a trusted identity provider. In this case, the application is Jenkins, which is a CI/CD tool that can integrate with Conjur using the Conjur Jenkins plugin. The plugin allows Jenkins to securely retrieve secrets from Conjur and inject them as environment variables into Jenkins pipelines or projects. The log entry indicates that the JWT sent by Jenkins was rejected by Conjur because of an SSL connection error. The error message says that the certificate chain of Jenkins could not be verified by Conjur, and that the certificate authority (CA) that signed the Jenkins certificate was unknown to Conjur. This means that the Jenkins certificate chain is not trusted by Conjur, and that Conjur does not have the CA certificate of Jenkins in its trust store. Therefore, Conjur cannot establish a secure and trusted connection with Jenkins, and cannot validate the JWT signature. To fix this problem, the Jenkins certificate chain needs to be trusted by Conjur. This can be done by copying

the CA certificate of Jenkins to the Conjur server, and adding it to the Conjur trust store. The Conjur trust store is a directory that contains the CA certificates of the trusted identity providers for the authn-jwt method. The Conjur server also needs to be restarted for the changes to take effect. References: Conjur Jenkins Plugin; Conjur JWT Authentication; Conjur Trust Store