



SECRETS MANAGER

CONJUR CLOUD INTEGRATION - TECHNICAL DOCUMENTATION TEMPLATE

DOCUMENT PURPOSE: TO PROVIDE THE CYBERARK DOCUMENTATION TEAM WITH THE RELEVANT INFORMATION ABOUT HOW TO WORK WITH AN INTEGRATION WITH CONJUR CLOUD.

Name of Product: Sweet Security

Version: 1

Date: February 26, 2025

CoNTACTS

Business Lead	Sarah Elkaim
Technical Contact	Orel Ben Ishay
Support	Orel Ben Ishay

REFERENCES

Marketplace link: [<add link>](#)

INTEGRATION OVERVIEW

Sweet Security and CyberArk partnered to bridge the gap between runtime visibility and secrets management for Kubernetes environments. The integration provides the seamless detection of unmanaged secrets and enables their onboarding into Conjur for centralized management, in addition to providing insight and visibility on managed secrets at runtime.

Why this integration is important:

CyberArk customers can manage secrets in Conjur but often lack visibility into their usage within Kubernetes clusters at runtime. Unused secrets increase the attack surface and need to be deprecated. In addition, unmanaged secrets, which are not visible in Conjur, pose significant risks as they can be exploited by attackers. These secrets require proactive rotation and management. Sweet solves these gaps.

KEY BENEFITS

1. Managed Secrets Insights:

- View managed secrets that are not in use, at runtime, enabling deprecation or deletion to reduce the attack surface and eliminate risk.

2. Unmanaged Secrets Detection:

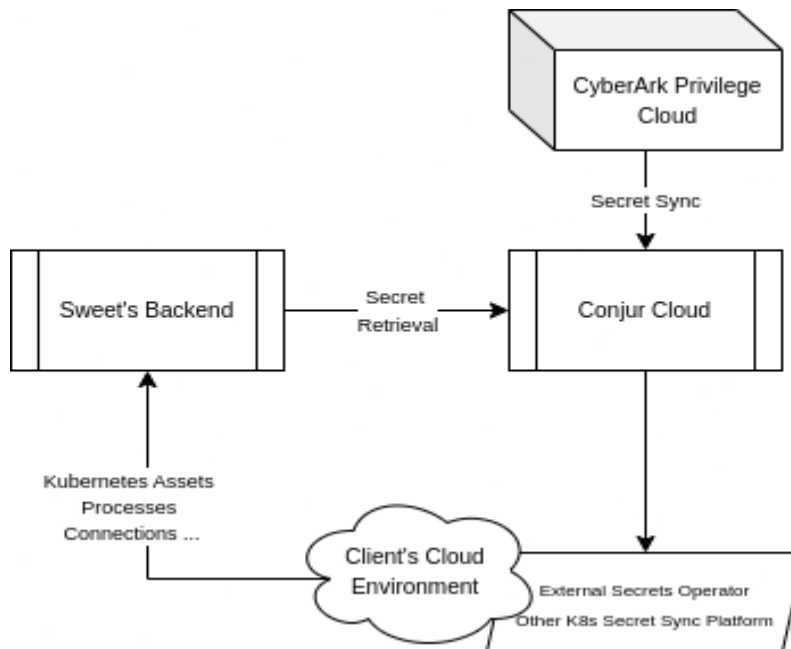
- Identify unmanaged secrets and their locations. Quickly and seamlessly onboard them directly into Conjur.

3. Runtime Insights for Unmanaged Secrets:

- Get detailed insight into the unmanaged secret, including if it's on a workload that is public facing/ has an inbound connection, or if the secret is in plain-text or both.

PRODUCT DESCRIPTION OF THE INTEGRATION

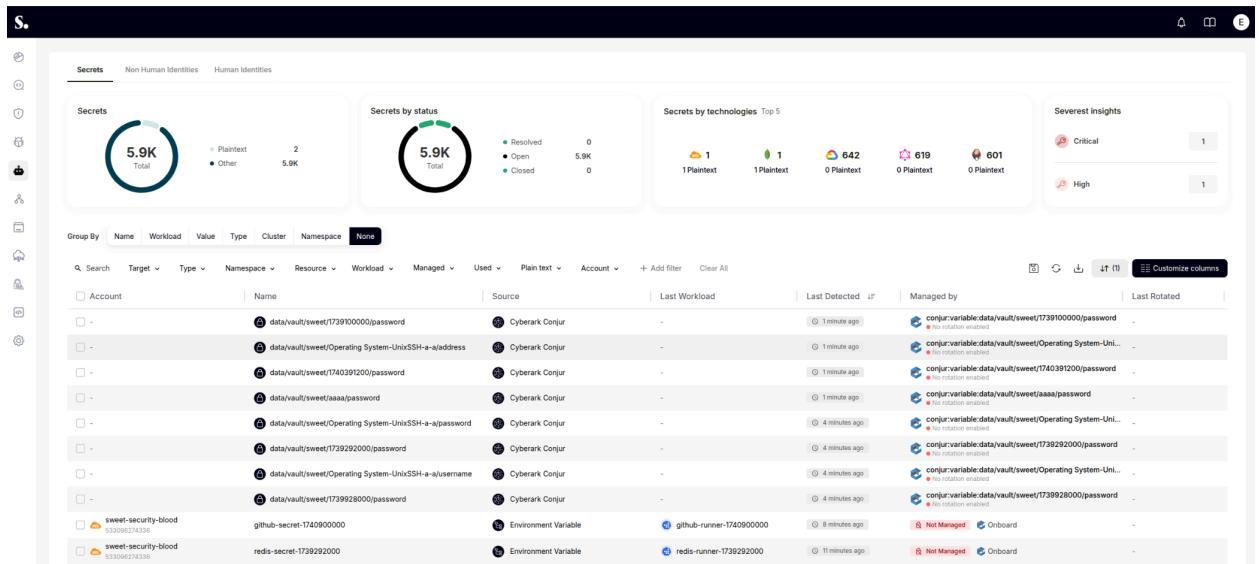
FLOW DIAGRAM



SECRETS MANAGER INTEGRATION

Secrets Manager is integrated into the Secrets Page, which is part of the Identities section within the product. The integration is present in key areas to ensure visibility and streamlined management:

- **Secrets Page (Identities Section):** The primary interface where both managed and unmanaged secrets are listed.
- **Managed By Column:** Displays whether a secret is managed using a Conjur identifier. This allows users to easily identify managed secrets and differentiate them from unmanaged ones.
- **Secrets List View:**
 - **Managed Secrets:** Clearly marked with a Conjur identifier, allowing users to view them directly within the Partner Product.
 - **Unmanaged Secrets:** Includes a link to the Secrets Manager portal to facilitate easy onboarding.



SECRET RETRIEVAL

Sweet will periodically query the Conjur API using the client's API key to fetch newly added secrets at fixed intervals.

In addition, Sweet will collect relevant Kubernetes assets, including:

- Pod manifests
- Process environment variables
- External Secrets

Sweet will then cross-reference the managed secrets from Conjur with the runtime data collected from Kubernetes to identify discrepancies and provide insights into the client's runtime environment and Conjur-managed secrets.

Below is a comparison between a managed secret from Conjur and an unmanaged secret discovered directly as an environment variable within a Kubernetes workload:

data/vault/sweet/1739928000/password	Cyberark Conjur	-	4 minutes ago	conjur:variable:data/vault/sweet/1739928000/password
				No rotation enabled
github-secret-1740900000	Environment Variable	github-runner-1740900000	8 minutes ago	Not Managed
				Onboard

To successfully integrate **Sweet** with **Conjur Cloud**, there are preparatory steps required on both the **Kubernetes** and **Conjur** environments, followed by providing the necessary details to Sweet for final configuration.

1. Kubernetes Environment Setup

Ensure the Kubernetes cluster is ready with the following components:

- **Runtime Workloads** These are the active applications and services running inside your Kubernetes cluster. Sweet's sensor will monitor these workloads for unmanaged or misused secrets.
 - **External Secrets Operator (ESO)** - Deploy the ESO to sync secrets from Conjur into Kubernetes as native Kubernetes secrets. Follow the ESO Getting Started Guide to configure it properly.
 - **Sweet Sensor** - Deploy Sweet's lightweight sensor to the Kubernetes environment. This sensor continuously monitors your runtime workloads for secret detection and policy enforcement. Details on deploying the sensor can be found [here](#).
-

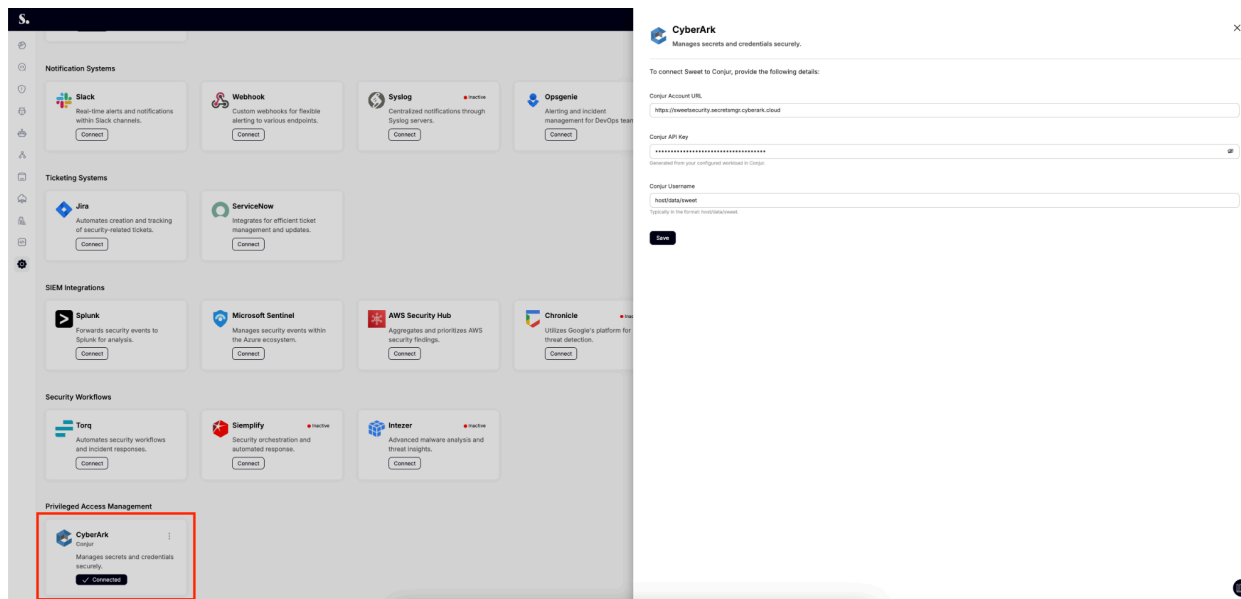
2. Conjur Cloud Preparation

- **Create a Workload in Conjur Cloud** Set up a workload (of type Other) within Conjur Cloud. Ensure the appropriate branches are selected—these branches will govern access to the relevant secrets.
 - **Generate Conjur API Key** After configuring the workload, generate the **API key** that Sweet will use for authentication.
-

3. Provide Integration Details to Sweet

To finalize the integration, input the following within the Sweet integration page under the **Cyberark** integration.

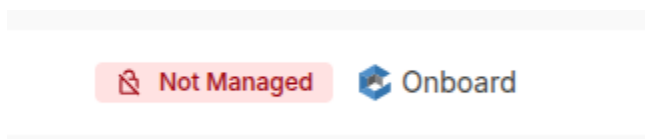
- **Conjur Account URL** Example: <https://sweetsecurity.secretsmgr.cyberark.cloud>
- **Conjur API Key** Generated from the workload you configured.
- **Conjur Username** Typically follows the format of your secrets prefix (e.g., host/data/sweet).



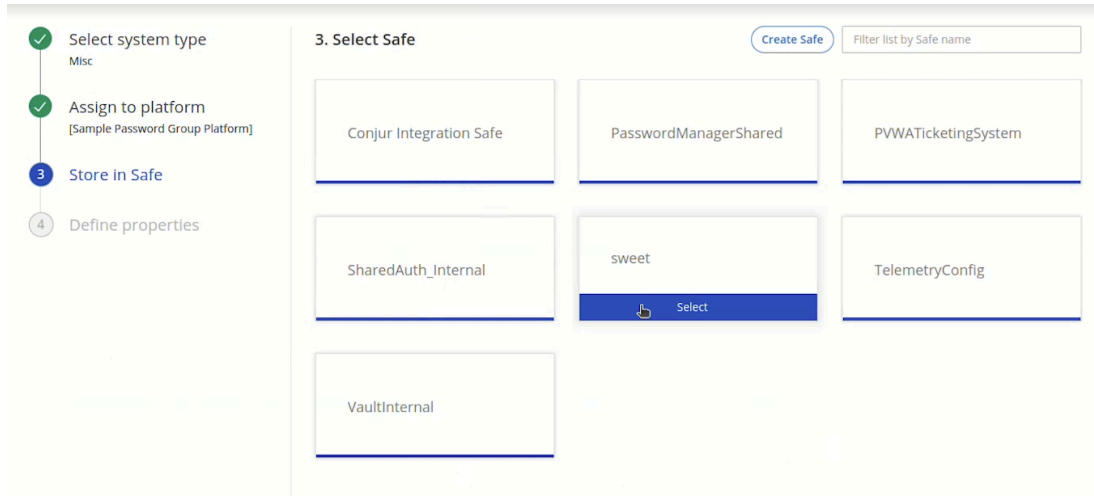
SECRETS ONBOARDING & SYNCHRONIZATION

To enable Sweet to recognize the managed secret and ensure it meets the criteria to be treated as a managed Kubernetes runtime secret, the customer should complete the following steps:

1. The customer should click the **Onboard** button next to the relevant unmanaged secret within Sweet's platform to initiate the onboarding process into Conjur via CyberArk's Privilege Cloud. This action will redirect the customer to Privilege Cloud to complete the onboarding by following [CyberArk's onboarding process](#).



The customer should ensure that, during onboarding, the selected **Conjur account** matches the account information previously provided to Sweet during the initial integration setup.



2. The customer should create an **ExternalSecret** resource within their Kubernetes cluster to synchronize the newly onboarded secret from Conjur into the cluster. This makes the secret available to runtime workloads. Implementation guidance can be found in the [External Secrets Operator documentation](#).

```
apiVersion: external-secrets.io/v1beta1
kind: ExternalSecret
metadata:
  name: conjur-external-secret
spec:
  refreshInterval: 10s
  secretStoreRef:
    name: conjur
    kind: SecretStore
  target:
    name: conjur-synced-secret
    creationPolicy: "Merge"
  data:
    - secretKey: password
      remoteRef:
        key: conjur:variable:<CONJUR_ONBOARDED_SECRET_ID>
```

3. The customer should update the relevant Kubernetes workload to consume the Kubernetes Secret created by the **ExternalSecret**. This is typically done by referencing the synced Secret in

the workload's deployment manifest (e.g., injecting it as an environment variable or mounting it as a volume). For additional guidance, the customer should refer to the official [Kubernetes documentation on using Secrets](#).

4. Once the workload has been updated to consume the synced Kubernetes Secret, Sweet will automatically detect the workload and the associated secret at runtime. Sweet will cross-reference this with Conjur's inventory and mark the runtime secret as **managed**.

Example: If a workload starts consuming a Kubernetes Secret synced from Conjur (e.g., gcp-secret), Sweet's sensor will detect that this secret is being used by the workload and confirm that it matches a secret managed by Conjur. As a result, Sweet will update its status and classify this runtime secret as managed, reducing the risk associated with unmanaged secrets.

Account	Name	Source	Last Workload	Managed by
 sweet-security-demo-production 727646487626	 graphql-secret-1738173601	 Environment Variable	 graphql-runner-1738173601	 conjur:variable:data/vault/sweet/1738173601/password No rotation enabled
 sweet-security-demo-production 727646487626	 gcp-secret-1738224001	 Environment Variable	 gcp-runner-1738224001	 conjur:variable:data/vault/sweet/1738224001/password No rotation enabled
 sweet-security-demo-production 727646487626	 mongo-secret-1738778400	 Environment Variable	 mongo-runner-1738778400	 conjur:variable:data/vault/sweet/1738778400/password No rotation enabled

INTEGRATION LIMITATIONS AND OTHER INFORMATION

There are no significant limitations to this integration. However, it is important to note that Sweet Security does not create secrets directly in Conjur on behalf of the user. Instead, Sweet offers a seamless experience by providing a one-click action to log into Conjur and initiate the secret onboarding process.