

CyberArk GitGuardian Integration Guide

v.1.1



Revision History:

Revision Number	Author	Summary
20240401-01	David Hisel	Version 1.0
20240418-01	David Hisel	Version 1.1 – Add Credential Provider

Table Of Contents



General Requirements	4
Linux Machine Configuration.....	5
Requirements	5
Setup The Frontend Proxy	5
GitGuardian Configuration	6
Requirements	6
Generate Personal Access Token.....	6
Create the GitGuardian Webhook.....	7
Install and Build the Integration Service	9
Configure The Integration Service.....	9
Create a startup script.....	11
Credential Provider Installation	12
Credential Provider Deamon Installation	12
Credential Provider SDK Installation	12
Install and Build the Integration Service with Credential Provider	13
Install the CyberArk GitGuardian Integration Host Platform.....	13
Configure The Integration Service.....	14
Create a Privilege Cloud Application	18
Create a startup script.....	19



General Requirements

Customers must have the following in order to implement this capability:

- ♦ Privilege Cloud Account
 - ♦ Safe created for storing Pending accounts
 - ♦ User that has access to the Pending safe with sufficient permissions to add accounts to the safe
 - ♦ CPM Plugin installed – CPM plugin can be downloaded from the CyberArk marketplace
- ♦ Linux Host – this is where the integration application will run
- ♦ PostgreSQL (or PostgreSQL compatible) database accessible by the Linux Machine
- ♦ Gitguardian Account
 - ♦ Personal Access Token
 - ♦ Webhook Authorization Token

To enable the **Credential Provider** feature, the customer must have the Credential Provider installed to the Linux host machine where the integration application is running.

Additional requirements to enable the Credential Provider feature:

- ♦ Install Credential Provider on Linux host
 - ♦ Credential Provider Service
 - ♦ Credential Provider SDK
- ♦ Privilege Cloud Account
 - ♦ “CyberArk GitGuardian Integration Host” Platform Installed – Download from the CyberArk marketplace.



Linux Machine Configuration

Requirements

- Git and Go lang installed
- Network configured to allow incoming GitGuardian webhook requests
- Front-end proxy installed, NGINX, to proxy pass to the integration and termination point for SSL traffic
 - Hostname resolves in DNS
 - SSL Certs created for the endpoint

Setup The Frontend Proxy

- 1) Login to a linux host where you will run the integration service
- 2) Install NGINX
- 3) Configure NGINX to be a frontend proxy; here is an example configuration where the hostname has been added to DNS as “webhook1.example.com” and the associated SSL certificates have been created and are available on the linux host for NGINX. Example config:

```
server {
    listen      443 ssl;
    server_name  webhook1.example.com;
    ssl_certificate ssl/webhook1.example.com/webhook1.example.com.cer;
    ssl_certificate_key ssl/webhook1.example.com/webhook1.example.com.key;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;
    ssl_ciphers  HIGH:!aNULL:!MD5;

    location / {
        proxy_pass http://localhost:9191;
        include proxy_params;
    }
}
```



GitGuardian Configuration

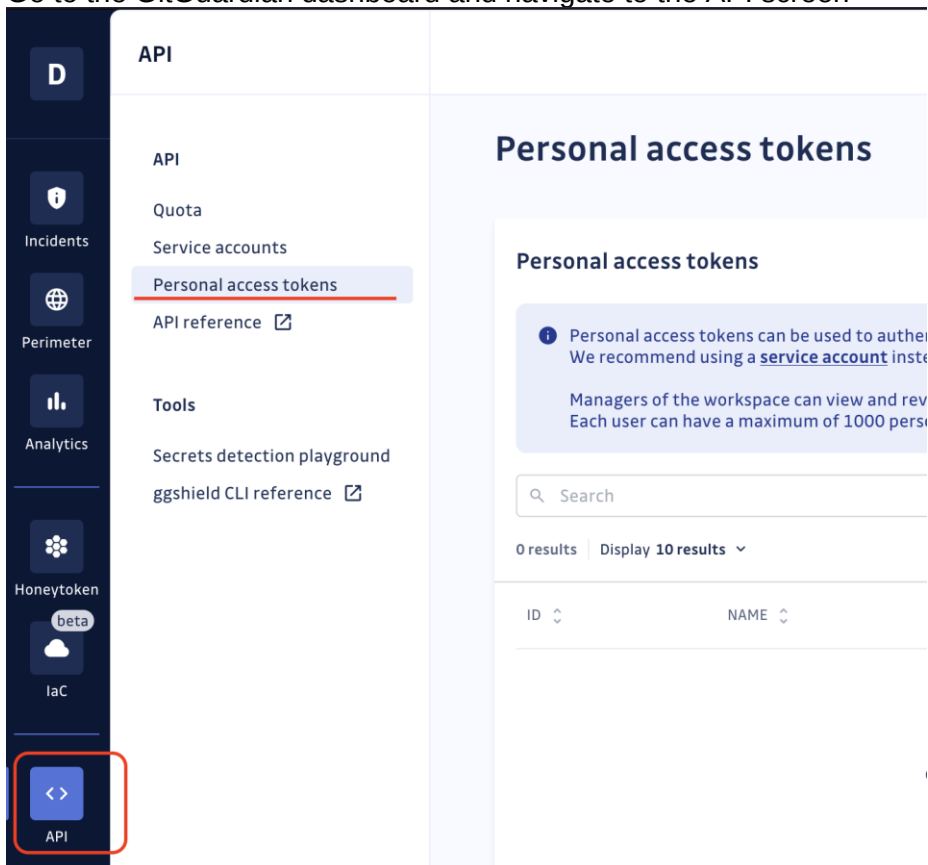
Requirements

- Endpoint defined for the integration, ex: <https://webhook1.example.com/>

Generate Personal Access Token

This is used when calling HMSL to avoid the daily limit of 5 requests per day; GitGuardian login url: <https://dashboard.gitguardian.com/>

- 1) Go to the GitGuardian dashboard and navigate to the API screen



- 2) Click on the Create Token button to create the PAT



- 3) Check the “Scan” checkbox, and click create token

New personal access token

Name *

cyberark-gg-integration-token

Expires

Never

Scope *

☒ scan	Secrets detection capability
☐ incidents	Full control of incidents
☐ incidents:share	Share, read and write incidents
☐ incidents:write	Read and write incidents
☐ incidents:read	Read incidents
☐ honeytokens	Full control of honeytokens
☐ honeytokens:write	Read and write honeytokens
☐ honeytokens:read	Read honeytokens
☐ members	Full control of members
☐ members:write	Read and write members
☐ members:read	Read members
☐ teams	Full control of teams
☐ teams:write	Read and write teams
☐ teams:read	Read teams
☐ audit_logs:read	Read audit logs
☐ api_tokens	Full control of API tokens
☐ api_tokens:write	Read and write API tokens
☐ api_tokens:read	Read API tokens

Cancel

Create token

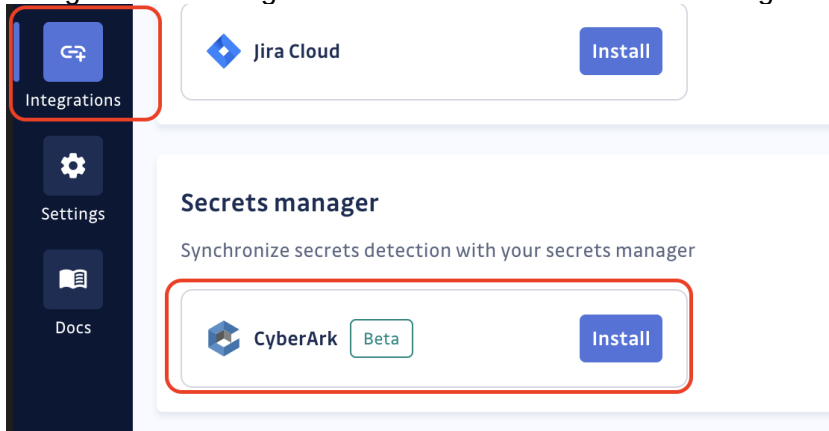
- a) Copy the token and save it for configuring the integration

Create the GitGuardian Webhook

This is used when calling HMSL to avoid the daily limit of 5 requests per day; GG login url:
<https://dashboard.gitguardian.com/>



- 1) Navigate to the Integrations and then to the Secrets Manager section



- 2) Click the install button and configure the integration – **Copy the webhook token** this will be used in the integration configuration.

[Integrations](#) > [CyberArk](#)



Configure your CyberArk integration

- 1 Connect GitGuardian with CyberArk to:
 - Automatically trigger a secrets rotation when one of your vaulted secrets is detected by GitGuardian
 - Add secrets detected by GitGuardian in your vault if they're not vaulted already
- Integrating with CyberArk requires to run a specific service between GitGuardian and your CyberArk vault. [Learn more in the documentation.](#)

⚠ Make sure to copy your signature token. You will need it to set up your integration with CyberArk, and won't be able to access it again.

Name *

priv cloud integration service

Content type

application/json

Instance URL *

https://gg-webhook.example.com/

Signature token *

abcdefghijklmnop

Submit

Webhook Token that the integration uses to authenticate the incoming GG requests.

This token is used in the integration configuration.



Install and Build the Integration Service

- 1) Create a directory to hold the integration code and change into the directory

```
mkdir work  
cd work
```

- 2) Clone the git repo to the machine

```
git clone https://github.com/conjurdemo/cyberark-gitguardian-hmsl-remediation-integration-service.git
```

- 3) Change into the repo directory

```
cd cyberark-gitguardian-hmsl-remediation-integration-service
```

- 4) Compile the binaries using make:

```
make build-all-bins
```

Configure The Integration Service

- 1) Obtain connection information for the PostgreSQL server

Example:

```
db user: DBUSER  
db pass: DBPASS  
db host: localhost  
db port: 26527  
db name: brimstone (database name)
```

Using these parameters would create the connection string:

```
-dburl "postgresql://DBUSER:DBPASS@localhost:26257/brimstone"
```

- 2) Obtain connection information for the Privilege Cloud account

Example:

```
ID Tenant ID: ID_TENANTID  
Privilege Cloud Tenant ID: PCLLOUD_TENANTID  
Privilege Cloud Service User: PCLLOUD_SERVICE_USER  
Privilege Cloud Service User Pass: PCLLOUD_SERVICE_USER_PASS  
Pending Safe name: PENDING_SAFENAME
```

Using these parameters to populate the service configuration configuration

```
-idtenanturl "https://ID_TENANTID.id.cyberark.cloud"  
-pcloudurl "https://PCLLOUD_TENANTID.privilegecloud.cyberark.cloud"  
-pamuser "PCLLOUD_SERVICE_USER"  
-pampass "PCLLOUD_SERVICE_USER_PASS"
```



```
-safename "PENDING_SAFENAME"
```

- 3) Determine the port that the service will listen on; this will be the port that the NGINX server is configured to proxy pass

Example:

```
listen on port 9191
```

Using this to populate the configuration parameter

```
-port "9191"
```

- 4) Generate an API key that the service will use to authenticate incoming requests with

Example:

```
generate a dev API key with value: dev123
```

Using this to populate the configuration environment variable

```
export BRIMSTONE_API_KEY="dev123"
```



Create a startup script

- 1) Using the configuration information gathered, fill in the appropriate fields in the example script, save the script in the “./bin” directory

```
#!/usr/bin/bash

# set the value of the API key for authn to brimstone
export BRIMSTONE_API_KEY="dev123"

# set this to the value of the GitGuardian API Token
export GG_API_TOKEN="abc123"

# set this to the value of the GitGuardian Webhook token presented
# when configuring the webhook
export GG_WEBHOOK_TOKEN="xxx"

./bin/brimstone -d \
-hmslurl "https://api.hasmysecretleaked.com" \
-hmslautype "hmsl" \
-ggapiurl "https://api.gitguardian.com" \
-ggapitokenvar "GG_API_TOKEN" \
-ggwebhooktokenvar="GG_WEBHOOK_TOKEN" \
-keyvar "BRIMSTONE_API_KEY" \
-dburl "postgresql://DBUSER:DBPASS@localhost:26257/brimstone" \
-port "9191" \
-identanturl "https://ID_TENANTID.id.cyberark.cloud" \
-pcloudurl "https://PCLOUD_TENANTID.privilegecloud.cyberark.cloud" \
-pamuser "PCLOUD_SERVICE_USER" \
-pampass "PCLOUD_SERVICE_USER_PASS" \
-safename "PENDING_SAFENAME" > ./bin/brimstone.log 2> ./bin/brimstone-err.log
```

- 2) Run the script to check that everything is configured properly

```
bash ./bin/local-start-script.sh
```

- 3) Check the logs to verify it is running properly

```
tail ./bin/brimstone.log
# Check log output

tail ./bin/brimstone-err.log
# Check log output
```



Credential Provider Installation

Install the Credential Provider on the same linux host where you will run the integration application. Follow this link and follow the instructions to install the CP:

[Credential Provider Documentation](#)

URL: https://docs.cyberark.com/credential-providers/latest/en/Content/CP+and+ASCP/lp_cp.htm

Credential Provider Deamon Installation

Installation of CP service:

<https://docs.cyberark.com/credential-providers/latest/en/Content/CP+and+ASCP/Installation-on-AIX-RHEL-CentOS.htm>

Credential Provider SDK Installation

Installation of SDK:

<https://docs.cyberark.com/credential-providers/latest/en/Content/CP%20and%20ASCP/Working-with-Application-Password-SDK.htm>

Configuring the SDK:

Note: where it says to copy the files into a /usr/lib* directory, copy the files, and do not symlink to the files.

<https://docs.cyberark.com/credential-providers/latest/en/Content/CP%20and%20ASCP/C-Application-Password-SDK.htm?tocpath=Developer%7CCredential%20Provider%7CApplication%20Password%20SDK.htm#tocpath=Developer%7CCredential%20Provider%7CApplication%20Password%20SDK%7C0#SetuptheCApplicationPasswordSDK>



Install and Build the Integration Service with Credential Provider

- 1) Create a directory to hold the integration code and change into the directory

```
mkdir work  
cd work
```

- 2) Clone the git repo to the machine

```
git clone https://github.com/conjurdemo/cyberark-gitguardian-hmsl-remediation-integration-service.git
```

- 3) Change into the repo directory

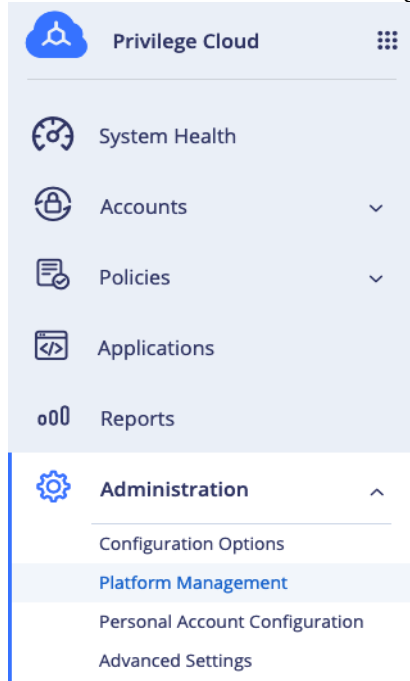
```
cd cyberark-gitguardian-hmsl-remediation-integration-service
```

- 4) Compile the binaries using make:

```
make build-all-bins
```

Install the CyberArk GitGuardian Integration Host Platform

- Download from the CyberArk marketplace
TBD: Where is the new platform definition coming from?
- Import into the Privilege Cloud account
 - Administration -> Platform Management -> Import platform

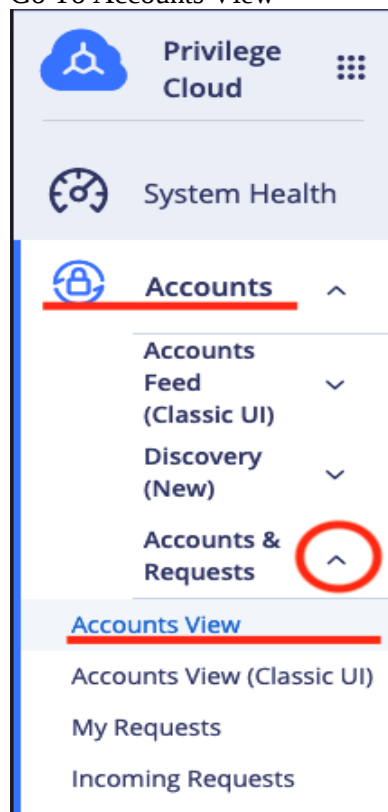


- The new platform should show up under “Applications”

		Verify password		Change password	
Platform Name		Periodic	Manual	Periodic	Manual
» Network Device (1)					
» Application (9)					
<u>CyberArk GitGuardian Integration Host</u>		-	-	-	✓

Configure The Integration Service

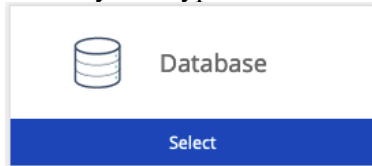
- Create account for database using PostgreSQL platform
 - Go To Accounts View



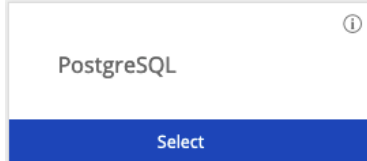
- Click on Add Account
- ### Accounts View



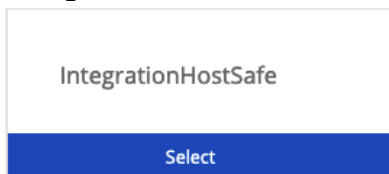
1. Select system type “Database”



2. Select platform “PostgreSQL”



3. Select safe where to add the account, for this example, we have created safename “IntegrationHostSafe” – This is the safe where the integration credentials will be stored



4. Fill in the properties for the database, use the values gathered from setting up the database step, then click save button

4. Account properties

Primary properties

Username

Address (optional)

Account name

Additional properties

DSN (ODBC) (optional)

Port (optional)

Database (optional)

Account management

☐ Allow automatic password management

Specify a reason for manual management request

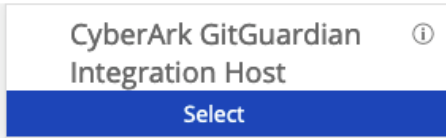


- Create account for integration service using CyberArk GitGuardian Integration Host platform

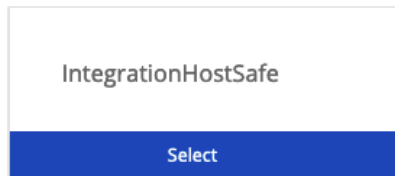
1. Select system type “Application”



2. Select platform “CyberArk GitGuardian Integration Host”



3. Select safe where to add the account, for this example, we have created safename “ExamplePendingSafe” – This is the safe where the integration credentials will be stored



4. Fill in the properties for the integration host, use the values gathered from setting up the linux host step, then click the save button



4. Account properties

Primary properties

Account name

Cybr-GG-Integration-Host

Additional properties

Integration Host API Key for clients to authenticate with

dev123

Integration Host Port to listen on

9191

GitGuardian API URL

https://api.gitguardian.com

GitGuardian API Token

my-gg-api-token

GitGuardian Webhook Token

my-gg-webhook-token

Privilege Cloud ID Tenant URL

https://EXAMPLE-TENANT-ID.id.cyberark.cloud

Privilege Cloud URL

https://EXAMPLE.privilegecloud.cyberark.cloud

Privilege Cloud PAM User

service_account_user@cyberark.cloud

Privilege Cloud PAM Password

my-pam-service-user-password

Privilege Cloud Pending Safe Name

ExamplePendingSafe

Account management

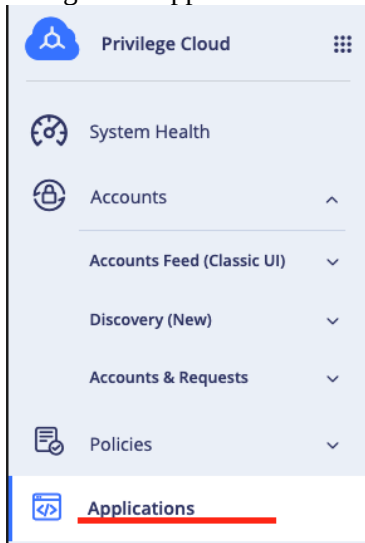
☒ Allow automatic password management

Specify a reason for manual management request

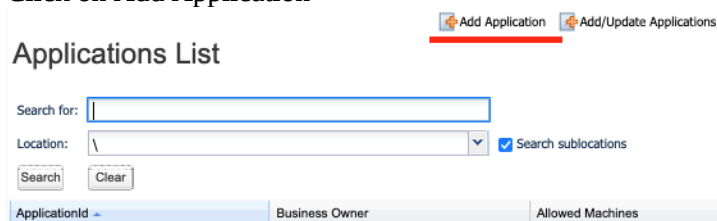
example account

Create a Privilege Cloud Application

- 1) Navigate to Applications



- 2) Click on Add Application



- 3) Add a name for the application, this will be used in the configuration in the startup script, placeholder in the startup script is INTEGRATION-HOST-APPLICATION-NAME, in the screenshot the value from the "Name:" field is what you will use in the startup script.

A screenshot of the 'Add Application' dialog box. The 'Name' field contains the text 'Integration-Host-Application'. The 'Description' field is empty. Below this is a section titled 'Business owner' with fields for 'First Name', 'Last Name', 'Email' (containing 'example1@example.com'), and 'Phone'. There is also a 'Location' dropdown menu set to '\Applications'. At the bottom, there are checkboxes for 'Access Permitted', 'Expiration Date', and 'Disabled'. The 'Add' and 'Cancel' buttons are at the bottom right.

Create a startup script

- 1) Using the configuration information gathered, fill in the appropriate fields in the example script, save the script in the “./bin” directory

```
#!/usr/bin/bash

./bin/brimstone-cp -d \
-safename "SAFENAME-WHERE-INTEGRATION-HOST-SETTINGS-ARE-STORED" \
-hostobjname "CYBR-GG-INTEGRATION-HOST-ACCOUNT-NAME" \
-dbobjname "POSTGRESQL-ACCOUNT-NAME" \
-appid "INTEGRATION-HOST-APPLICATION-NAME" </dev/null > ./bin/brimstone-cp.log 2> ./bin/brimstone-cp-err.log
```

- 2) Run the script to check that everything is configured properly

```
bash ./bin/local-start-script-cp.sh
```

- 3) Check the logs to verify it is running properly

```
tail ./bin/brimstone-cp.log
# Check log output

tail ./bin/brimstone-err-cp.log
# Check log output
```

