

Deploy Falcon Container Sensor for Linux Embedded in a Container Image

Last updated: Feb. 24, 2025

Overview

The Falcon Container sensor for Linux extends runtime security to container workloads where you can't deploy the kernel-based Falcon sensor for Linux. The Falcon Container sensor for Linux secures your workloads by running as an unprivileged container in user space with no code running in the kernel of the worker node OS.

If you are using one of these environments, follow the tailored deployment instructions at the links provided:

- [Deploy Falcon Container Sensor for Linux on ECS Fargate \[documentation/page/a5c297cc/deploy-falcon-container-sensor-for-linux-on-ecs-fargate\]](#)
- [Deploy Falcon Container Sensor for Linux on Google Cloud Run \[documentation/page/p6af9353/deploy-falcon-container-sensor-for-linux-on-google-cloud-run\]](#)
- [Deploy Falcon Container Sensor for Linux on Azure Container Instances \[documentation/page/ndf35434/deploy-falcon-container-sensor-for-linux-on-azure-container-instances-0\]](#)
- [Deploy Falcon Container Sensor for Linux on Azure Container Apps \[documentation/page/caf5a7b6/deploy-falcon-container-sensor-for-linux-on-azure-container-apps\]](#)

Otherwise, follow the instructions in this document to deploy the Falcon Container sensor for Linux in other environments that support deployments based on container images.

Note: The tailored deployment options provide more container visibility in the Falcon console than this more generic deployment method. This is because the tailored deployments contain additional parameters that fetch specific container metadata available in those environments.

Architecture

The Falcon Container sensor for Linux runs inside each application container. It tracks activity in the application containers and sends telemetry data to the CrowdStrike cloud. While the Falcon Container sensor for Linux is scoped to the container where it runs, its functionality is otherwise similar to that of the kernel-based Falcon sensor for Linux. In particular, it generates detections and performs prevention operations for activity in those application containers.

Deploying the Falcon Container sensor for Linux requires modification of the application container image. The Falcon Container sensor image contains a Falcon utility that supports patching the application container image with Falcon Container sensor for Linux and its related dependencies.

The Falcon Container has 2 components:

- **Falcon Container sensor for Linux:** At runtime, the Falcon Container sensor for Linux uses unique technology to launch and run inside the application container.
- **Falcon utility:** The Falcon utility runs offline and uses application container image to generate a new container image patched with the Falcon Container sensor for Linux and its related dependencies. The Falcon utility also sets the Falcon entry point as the container entry point. The Falcon utility can run on any host that has a Docker engine running.

Installation workflow summary

- [Step 1: Create an API client key \[documentation/page/k58f1a5e/deploy-falcon-container-sensor-for-linux-embedded-in-a-container-image#m0969032\]](#)
- [Step 2: Get your CrowdStrike CID with checksum \[documentation/page/k58f1a5e/deploy-falcon-container-sensor-for-linux-embedded-in-a-container-image#icc6bd8a\]](#)
- [Step 3: Retrieve the sensor image and push to your repository \[documentation/page/k58f1a5e/deploy-falcon-container-sensor-for-linux-embedded-in-a-container-image#cdaf9149\]](#)

Step 4: Run the Falcon utility to build a new image [/documentation/page/k58f1a5e/deploy-falcon-container-sensor-for-linux-embedded-in-a-container-

- [image#w496eec2](#)

Option 1: Run the Falcon utility within the Falcon Container [/documentation/page/k58f1a5e/deploy-falcon-container-sensor-for-linux-embedded-in-

- [a-container-image#r2ed1c08](#)

Option 2: Run the Falcon utility on the host as a standalone binary [/documentation/page/k58f1a5e/deploy-falcon-container-sensor-for-linux-

- [embedded-in-a-container-image#l63fc749](#)

Step 5: Push the new image to the registry [/documentation/page/k58f1a5e/deploy-falcon-container-sensor-for-linux-embedded-in-a-container-

- [image#g34ca869](#)

Step 6: Deploy Falcon Container sensor [/documentation/page/k58f1a5e/deploy-falcon-container-sensor-for-linux-embedded-in-a-container-

- [image#c0a98a84](#)

Step 7: Verify the sensor deployment [/documentation/page/k58f1a5e/deploy-falcon-container-sensor-for-linux-embedded-in-a-container-

- [image#gcdb3398](#)


Step 1: Create an API client key

The API client ID is used to retrieve the Falcon sensor image. Create a CrowdStrike API client ID with the required API scopes.

1. In the Falcon console, go to [Support and resources > Resources and tools > API clients and keys](#) [/api-clients-and-keys].
2. Click **Create API client**.
3. Enter a client name and description.
4. Select these scopes:
 - **Falcon Images Download: Read**
 - **Sensor Download: Read**
5. Click **Create**.
6. In the **API client created** dialog, copy the **Client ID** and **Secret** to a password management or secret management service.
7. Set your CrowdStrike client ID and secret as environment variables:

```
export FALCON_CLIENT_ID=<your_falcon_client_id>
export FALCON_CLIENT_SECRET=<your_falcon_client_secret>
```

Step 2: Get your CrowdStrike CID with checksum

1. In the Falcon console, go to [Host setup and management > Deploy > Sensor downloads](#) [/hosts/sensor-downloads].
2. On the **Sensor downloads** page, in the **How to install** section, locate your customer ID.
3. Click copy  to copy your CID with checksum to the clipboard.
4. Set the CID with checksum as an environment variable:

```
export FALCON_CID=<your_cid_with_checksum>
```

Step 3: Retrieve the sensor image and push to your repository

Run the container pull script. The script uses your CrowdStrike API client ID and secret to fetch CrowdStrike registry credentials, set up your local Docker client, and pull the latest sensor image.

Important: Falcon sensor images are continuously assessed for vulnerabilities and malware before and after release.

1. Run the Falcon Container sensor pull script to fetch the latest image and save the sensor URI as \$LATESTSENSOR:

```
export LATESTSENSOR=$(bash <(curl -Ls https://github.com/CrowdStrike/falcon-
scripts/releases/latest/download/falcon-container-sensor-pull.sh) -t falcon-container | tail -1) &&
echo $LATESTSENSOR
```

2. Create the repository for the Falcon Container sensor image in your registry and set it as a variable:

```
export MY_REPO=<your_image_repo>
```

3. Tag the sensor image:

```
docker tag $LATESTSENSOR $MY_REPO:latest
```

4. Push the sensor image to your repo:

```
docker push $MY_REPO:latest
```

Step 4: Run the Falcon utility to build a new image

The Falcon utility, `falconutil`, is packaged within the Falcon Container sensor image. Run this utility locally to patch your application container image with the Falcon Container sensor for Linux. You can also run the Falcon utility as part of your CI/CD process.

Prerequisite: Generate Base64-encoded registry credentials

The Falcon utility pulls both the application container image and Falcon Container sensor image from the image registry. It uses Base64-encoded credentials from Docker's `config.json` to authenticate the image registries.

Example:

For Microsoft Entra ID (formerly Azure Active Directory), one of the methods to generate the Base64-encoded credentials is by authenticating to the registry directly using an individual login. For info, see the Microsoft article [Individual login with Microsoft Entra ID \[https://learn.microsoft.com/en-us/azure/container-registry/container-registry-authentication?tabs=azure-cli#individual-login-with-azure-ad\]](https://learn.microsoft.com/en-us/azure/container-registry/container-registry-authentication?tabs=azure-cli#individual-login-with-azure-ad).

Run the utility

To run the utility, use one of these options:

- Option 1: Run the Falcon utility within the Falcon Container [documentation/page/k58f1a5e/deploy-falcon-container-sensor-for-linux-embedded-in-a-container-image#r2ed1c08]
- Option 2: Run the Falcon utility on the host as a standalone binary [documentation/page/k58f1a5e/deploy-falcon-container-sensor-for-linux-embedded-in-a-container-image#l63fc749]

You should only run the Falcon utility for those application container images on which you want the Falcon Container sensor for Linux to be injected.

Option 1: Run the Falcon utility within the Falcon Container

```
docker run --user 0:0 \
-v ${HOME}/.docker/config.json:/root/.docker/config.json \
-v /var/run/docker.sock:/var/run/docker.sock \
--rm $MY_REPO:latest \
falconutil patch-image\
--source-image-uri <source_image_uri> \
--target-image-uri <target_image_uri> \
--falcon-image-uri $MY_REPO:latest \
--cid <your_cid_with_checksum> \
--falconctl-opts <falconctl_options> \
--image-pull-policy <Always|IfNotPresent> \
--container <container_name>
```

Flag descriptions

Flag	Flag Required?	Description
--rm	Optional	Remove the container after it exits

--source-image-uri	Required	Application container image URI to be patched
--target-image-uri	Required	Expected URI for the target image
--falcon-image-uri	Required	Falcon Container sensor image URI
--cid	Required	Customer ID to use
--falconctl-opts	Optional	All falconctl options in a single string. For info, see Falconctl configuration options [documentation/page/dde40c99/configuration-options-for-falcon-container-sensor#wc4ab0be]
--image-pull-policy	Optional	Image pull policy for application container and Falcon Container sensor image. Supported values: Always, IfNotPresent. The default is Always.
--container	Optional	Container name. If set, this value can be used to identify the container in the Falcon console.

Notes:

- Mounting config.json is required when pulling images from registries that require authentication.
- Falcon utility uses Docker APIs to interact with the Docker daemon. Mounting /var/run/docker.sock is required to enable this interaction.

Option 2: Run the Falcon utility on the host as a standalone binary

Note: This option is only available for Linux hosts.

1. Copy the Falcon utility from the Falcon Container image:

- id=\$(docker create \$MY_REPO:latest)
- docker cp \$id:/usr/bin/falconutil /tmp
- docker rm -v \$id

2. Run the Falcon utility on the host as a standalone binary:

```
/tmp/falconutil patch-image\  
--source-image-uri <source_image_uri> \  
--target-image-uri <target_image_uri> \  
--falcon-image-uri $MY_REPO:latest \  
--cid <your_cid_with_checksum> \  
--falconctl-opts <falconctl_options> \  
--image-pull-policy <Always|IfNotPresent> \  
--container <container_name>
```

Flag descriptions

Flag	Flag Required?	Description
--source-image-uri	Required	Application container image URI to be patched
--target-image-uri	Required	Expected URI for the target image
--falcon-image-uri	Required	Falcon Container sensor image URI

--cid	Required	Customer ID to use
--falconctl-opts	Optional	All falconctl options in a single string. For info, see Falconctl configuration options [documentation/page/dde40c99/configuration-options-for-falcon-container-sensor#wc4ab0be]
--image-pull-policy	Optional	Image pull policy for application container and Falcon Container sensor image
--container	Optional	Container name. If set, this value can be used to identify the container on the Falcon console.

Step 5: Push the new image to the registry

After you build the new image using Falcon utility, push the image to the registry:

```
docker push <target_image_uri>
```

Step 6: Deploy Falcon Container sensor

After you push the new image to the registry, create a container with this image. The container starts with a modified entrypoint that starts the Falcon Container sensor for Linux followed by the user-defined entrypoint.

Note: The container name you passed to the --container flag in [Step 4: Run the Falcon utility to build a new image](#) [documentation/page/k58f1a5e/deploy-falcon-container-sensor-for-linux-embedded-in-a-container-image#w496eec2] must match the container name used at the time of deployment.

You can set these environment variables when deploying the container:

Environment variable	Description
CS_CONTAINER	<p>Container name</p> <p>It is equivalent to the --container flag in Step 4: Run the Falcon utility to build a new image [documentation/page/k58f1a5e/deploy-falcon-container-sensor-for-linux-embedded-in-a-container-image#w496eec2]</p> <p>The environment variable overrides the value set by the --container flag.</p>
CS_APP_NAME	<p>Application name</p> <p>This environment variable is a platform-agnostic parameter. It can be used to group related containers that are part of a business use case.</p> <p>The containers can be filtered on the Falcon console based on Application name.</p>

Step 7: Verify the sensor deployment

After you deploy the Falcon Container sensor for Linux to your environment, you should verify that it has been assigned an agent ID (AID) by the CrowdStrike cloud.

- To get the container's running processes, run this command:

```
ps -aef | grep falcon-sensor
```

Output similar to this shows that the Falcon processes are running:

```
/opt/CrowdStrike/rootfs/lib64/ld-linux-x86-64.so.2 --library-path
/opt/CrowdStrike/rootfs/lib64
/opt/CrowdStrike/rootfs/bin/falcon-sensor
```

2. Verify that the sensor is connected to the cloud by fetching the AID. Run this command:

```
/opt/CrowdStrike/rootfs/bin/falconctl -g --aid
```

When the sensor connects to the CrowdStrike cloud, it's assigned an AID, for example 85ae98xxxxxd9a8f2. If the output displays an AID, your sensor is running correctly.