Vulnerability Analysis Plugin for Jenkins

Version 1.6.0.0

August 1, 2019

Qualys, Inc.
919 E Hillsdale Blvd
4th Floor
Foster City, CA 94404
1 (650) 801 6100

Table of Contents

About this Guide

Welcome to Qualys Container Security! We'll help you get acquainted with the Qualys solutions for securing your Container environments like Images, Containers and Docker Hosts using the Qualys Cloud Security Platform.

About Qualys

Qualys, Inc. (NASDAQ: QLYS) is a pioneer and leading provider of cloud-based security and compliance solutions. The Qualys Cloud Platform and its integrated apps help businesses simplify security operations and lower the cost of compliance by delivering critical security intelligence on demand and automating the full spectrum of auditing, compliance and protection for IT systems and web applications.

Founded in 1999, Qualys has established strategic partnerships with leading managed service providers and consulting organizations including Accenture, BT, Cognizant Technology Solutions, Deutsche Telekom, Fujitsu, HCL, HP Enterprise, IBM, Infosys, NTT, Optiv, SecureWorks, Tata Communications, Verizon and Wipro. The company is also founding member of the Cloud Security Alliance (CSA). For more information, please visit www.qualys.com

Qualys Support

Qualys is committed to providing you with the most thorough support. Through online documentation, telephone help, and direct email support, Qualys ensures that your questions will be answered in the fastest time possible. We support you 7 days a week, 24 hours a day. Access online support information at www.qualys.com/support/.

About Container Security Documentation

This document provides information about using the Qualys Vulnerability Analysis Plugin for Jenkins.

For information on using the Container Security UI to monitor vulnerabilities in Images, Containers, and Registries, refer to the Qualys Container Security User Guide.

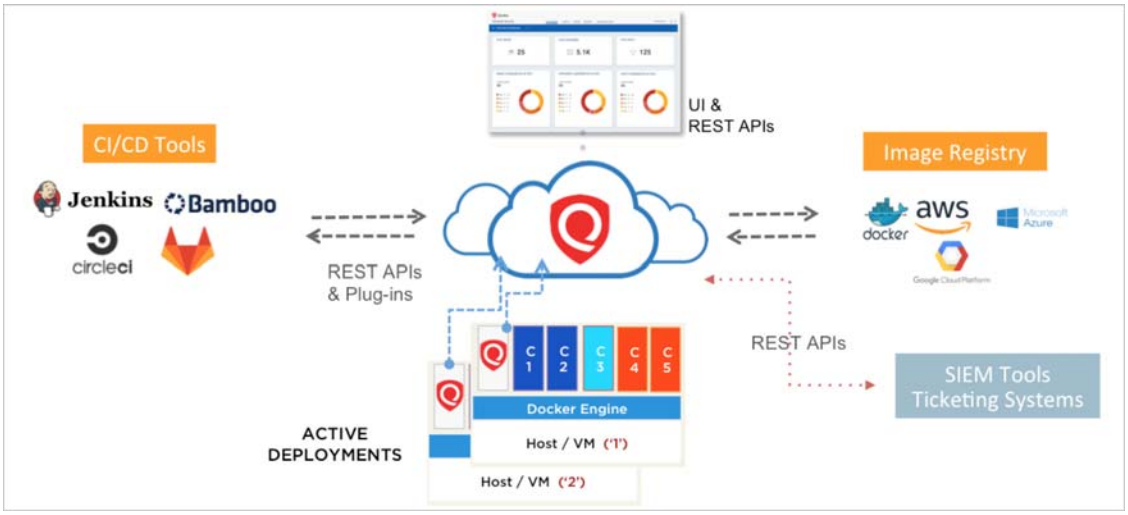
For information on deploying the sensor on MAC, CoreOS, and various orchestrators and cloud environments, refer to the Qualys Container Sensor Deployment Guide.

For information on using the Container Security API, refer to the Qualys Container Security API Guide.

For information on the Bamboo plugin, see Qualys Vulnerability Analysis Plugin for Bamboo.

Container Security Overview

Qualys Container Security provides discovery, tracking, and continuously protecting container environments. This addresses vulnerability management for images and containers in their DevOps pipeline and deployments across cloud and on-premise environments.



With this version, Qualys Container Security supports

- Discovery, inventory, and near-real time tracking of container environments

- Vulnerability analysis for images and containers

- Vulnerability analysis for registries

- Integration with CI/CD pipeline using APIs (DevOps flow)

- Uses new 'Container Sensor' – providing native container support, distributed as docker image

Upon installation, the sensor does automatic discovery of Images and Containers on the deployed host, provides a vulnerability analysis of them, and additionally it monitors and reports on the docker related events on the host. The sensor lists and scans registries for vulnerable images. The sensor container runs in non-privileged mode. It requires a persistent storage for storing and caching files.

Currently, the sensor only scans Images and Containers. For getting a vulnerability posture on the Host, you would require Qualys Cloud Agents or a scan through Qualys Virtual Scanner Appliance.

What data does Container Security collect?

The Qualys Container Security sensor fetches the following information about Images and Containers in your environment:

- **Inventory of Images and Containers** in your environment from commands such as **`docker ps`** that lists all containers.

- **Metadata information** about Images and Containers from commands such as **`docker inspect`** and **`docker info`** that fetches low level information on docker objects.

- **Event information** about Images and Containers from the docker host for docker events like created, started, killed, push, pull, etc.

- **Vulnerabilities** found on Images and Containers. This is the output of the vulnerability management manifests run for identifying vulnerability information in Images and Containers. This is primarily software package listing, services running, ports, etc.

For example, package manager outputs like **`rpm -qa`**, **`npm`**. This is supported across various Linux distributions (CentOS, Ubuntu, CoreOS, etc) and across images like Python, NodeJS, Ruby, and so on.

About the Vulnerability Analysis Plugin for Jenkins

Qualys Container Security provides a plugin for Jenkins to get the security posture for the docker images built via the tool. The plugin can be configured to fail or pass the docker image builds based on the vulnerabilities detected.

Getting started

Follow the steps to get started with Vulnerability Analysis Plugin for Jenkins.

What you'll need

- A valid Qualys subscription with the Container Security application activated.

- Access to Qualys Container Security application API endpoint from your build host.

- Requires the container sensor for CI/CD environment to be installed on the jenkins build host. Refer to Qualys Container Security Sensor Deployment Guide for instructions on installing the container cicd sensor. You must pass the following parameter while deploying the sensor for CI/CD environment `--cicd-deployed-sensor or -c`.

- Internet connection for slave to be able to connect to the Qualys Cloud Platform. Install sensor with proxy option if slave is running behind proxy.

- The Jenkins master and slave nodes should have an open connection to the Qualys Cloud Platform in order to get data from the Qualys Cloud Platform for vulnerability reporting.

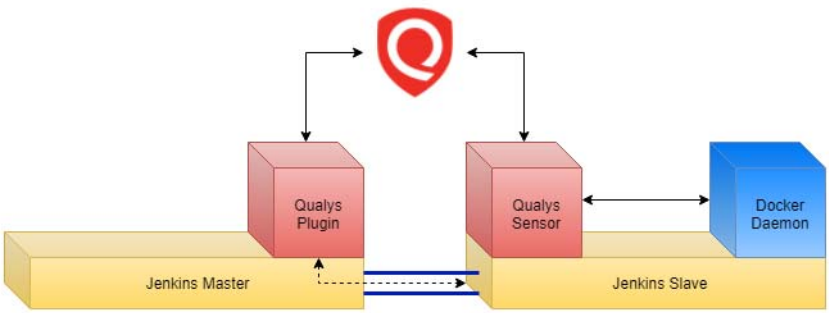
Jenkins plugin automatically tags images built out of CI/CD pipeline with the tag **qualys_scan_target:<image-id>** to mark them for scanning and only those images are scanned for vulnerabilities. Once the scanning is over, Qualys Container Sensor will remove the tag. However, if an image has no other tag applied to it other than 'qualys_scan_target:<image-id>', the sensor will retain the tag to avoid removal of the image from the host.

Note: Qualys Container Security does not support scanning images in Docker-in-Docker setup as the sensor cannot listen to events generated by the inner Docker.

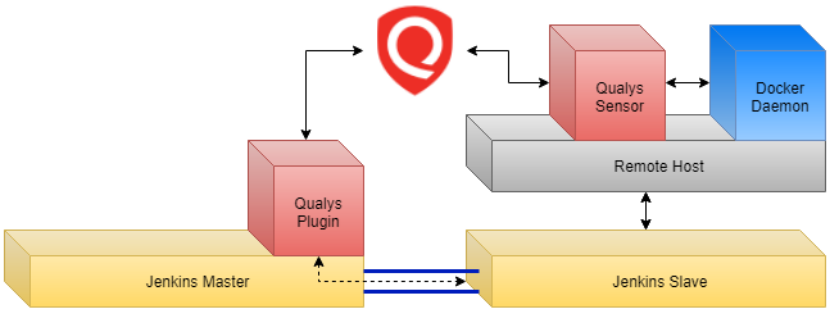
Recommended setup for master-slave deployment

Vulnerability Analysis Plugin for Jenkins should be deployed on the Jenkins master. Qualys Container Security Sensor should be installed where the docker daemon is running. If the docker daemon is running on Jenkins slave, install the Sensor on Jenkins slave. If the docker daemon is running on a remote host, install the sensor over there.

Following figure shows the docker daemon running on Jenkins salve.



Following figure shows the docker daemon running on a remote host.



Install the Plugin

1) On Qualys Cloud Platform, go to Configurations > Integrations to download the Jenkins plugin.

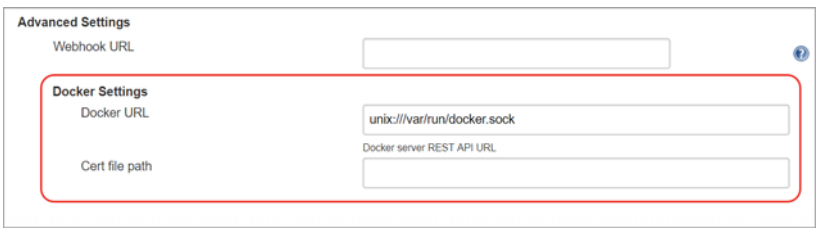
2) Upload the plugin to your Jenkins tool. Go to Manage Jenkins > Manage Plugins > Advanced > Upload Plugin, then restart it.

Scanning CI/CD images

Configure the docker URL / socket path for the plugin to automatically tag CI/CD images with 'qualys_scan_target:<image-id>'.

Docker URL: Docker REST API URL / Docker socket path. Only unix:/// and tcp:// protocols allowed.

Cert File Path (opional): If you are using remote server enabled https, you can provide a specific folder location which contains the files ca.pem, cert.pem and key.pem. For example, /var/jenkins_home/certs.



Note: A Job Specific (local) configuration will use the Docker URL and Cert File Path configured in global configuration for tagging CI/CD images.

Docker URLs (unix socket or TCP) to be used in various docker deployment scenarios

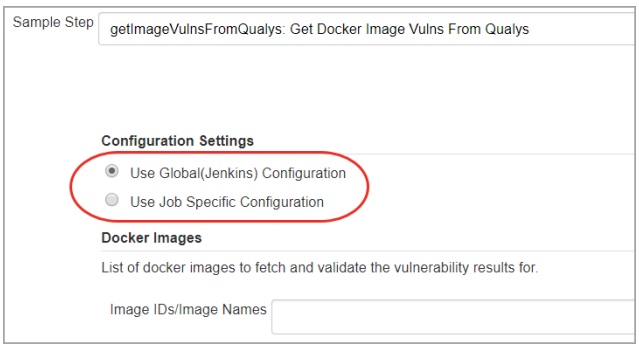
Deployment scenario	Sensor location	Docker URL to be used
Job executed by Jenkins master AND Docker host == Jenkins master	Jenkins master	UNIX unix:///var/run/docker.sock
Job executed by Jenkins master AND Docker Host == Remote docker host (any machine other than Jenkins master or slave)	Remote docker host	TCP path of the Remote Docker host: tcp://<ip_of_RDH>:<port> For example, tcp://10.115.67.61:2375
Job executed by Jenkins slave AND Docker host == Jenkins slave	Jenkins slave	UNIX unix:///var/run/docker.sock
Job executed by Jenkins slave AND Docker Host == Remote docker host (any machine other than Jenkins master or slave)	Remote docker host	TCP path of the Remote Docker host: tcp://<ip_of_RDH>:<port> For example, tcp://10.115.67.61:2375

Start Using the Plugin

Qualys recommends to set up the Jenkins Plugin after the docker image is built, and before the image is pushed to the registry. Ensure that you do not delete the image before the plugin setup is complete.

While setting up the plugin you can either provide **a global configuration or a job specific configuration**. Global configuration can be set once and used for multiple projects: both Pipeline and Freestyle.

To set a global configuration, go to Manage Jenkins > Configure System, then scroll down to the Qualys Container Security section, and provide the configuration details listed below.



If you want to set a job specific configuration, in the Snippet Generator, select "getImageVulnsFromQualys - Get Image Vulns From Qualys", and then select the Use Job Specific Configuration option.

Note: Selecting the "Use Global(Jenkins) Configuration" option here will let the job use the global configuration you have set under Manage Jenkins > Configure System > Qualys Container Security.

See Configuration Details

This plugin provides a build step and a post-build action. You can use it with pipeline type projects (for CI/CD pipeline) as well as freestyle projects. We'll describe both in the sections that follow.

- Pipeline Project

- Freestyle Project

Define docker image Ids

In the plugin configuration there is a field called image IDs/Image Names. Set this to the docker image Ids or names you want to report on. The plugin will only pull a report for the image Ids/names you specify.

Enter a single string value like imageIds: 'a1b2c3d4e5f6' or a comma-separated list like imageIds: 'a1b2c3d4e5f6,abcdef123456'. Specify an image name in the format `repo:tag`.

If you provide an image name, the plugin fetches the corresponding image ID. The plugin tries to fetch the image ID using the docker socket path configured in global configuration. If your docker host is running locally to build tool/agent, the docker socket path is unix:///var/run/docker.sock; whereas if your docker host is running remotely, the docker socket path is the TCP URL to the remote docker host. See Scanning CI/CD images.

You can also provide image ids through an environment variable. Get the image ids of the images programmatically created in earlier stages of the build and provide these ids in the 'imageIds' argument. For example, in pipeline script, you can get the image ids by executing shell script and store it in environment variable. And then use the same environment variable in 'ImageIds' argument to provide the image ids.

Pipeline Project

With pipeline projects, you provide the docker image Id(s) to the plugin via a command argument. Use the Snippet Generator to generate this command, and then copy/paste it into your pipeline script (Jenkinsfile).

What are the steps?

- Use a specific tag to build a docker image you wish to scan for vulnerabilities.

- Use that tag to get the ID of the docker image and then store that image ID into an environment variable.

- Provide that environment variable to the Qualys Jenkins Plugin as input.

Sample Pipeline script

```
stage('Build docker image') {
            //Build the docker image with a tag (qualys:sample in
this case)
            steps {
                dir("dockerbuild") {
                    sh "docker build -t qualys:sample . >
docker_output"
                }
            }
        }

        stage('Get Image id') {
            //Use the same repo:tag (qualys:sample in this case)
combination with the grep command to get the same image id and save
the image id in an environment variable
            steps {
                script {
                   def IMAGE_ID = sh(script: "docker images | grep
-E '^qualys.*sample' | head -1 | awk '{print \$3}'", returnStdout:
true).trim()
                    env.IMAGE_ID = IMAGE_ID
                }
```

```
            }
        }

        stage('Get Image Vulns - Qualys Plugin') {
             //Use the same environment variable(env.IMAGE_ID) as
an input to Qualys Plugin's step
            steps {
               getImageVulnsFromQualys useGlobalConfig:true,
imageIds: env.IMAGE_ID
            }
        }
```

Freestyle Project

You'll provide the docker image IDs/Image Names on the plugin configuration page. When Jenkins executes the post-build steps the plugin will only pull a report for the image Ids/names you've specified. You can also provide image ids through an environment variable.

In Post-Build Actions, select "Get docker image vulnerabilities from Qualys". This will open a form similar to the one shown for pipeline projects. Provide configuration details and test the connection to make sure it's successful. See How to configure?

Set the IMAGE_ID environment variable to the docker image Ids you want to report on. IMAGE_ID can be a single string value like 'a1b2c3d4e5f6' or a comma-separated list like 'a1b2c3d4e5f6,abcdef123456'.

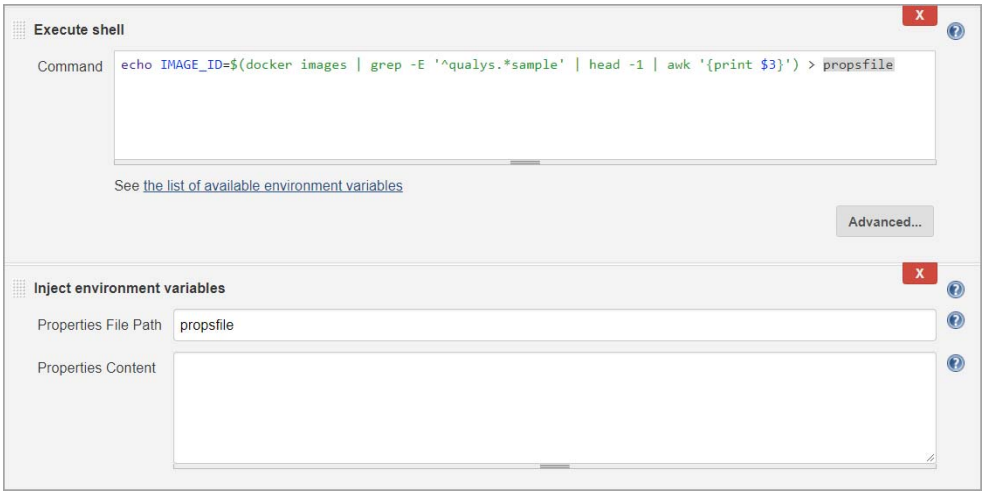
Note: The variable name must be defined correctly or the plugin will not work.

What are the steps?

- Use a specific tag to build a docker image you wish to scan for vulnerabilities.

- Add a build step called "Execute shell" in which write a shell script to use that tag to get the ID of the docker image and then store that image ID in the format: `IMAGE_ID=<image_id>` into a file`.`

- Add the "Inject environment variables" step in which provide the filename of the file in which you stored the image ID.



Using the WebHook

You can forward Jenkins job results to a WebHook URL.

You can set a global WebHook URL under Manage Jenkins > Configure System > Qualys Container Security, or a Job Specific WebHook URL under Snippet Generator by selecting “getImageVulnsFromQualys - Get Image Vulns From Qualys”, and then clicking Advanced.

Note: WebHook URL specified under Snippet Generator, for a particular project, will always take preference over the global WebHook URL specified under Manage Jenkins > Configure System > Qualys Container Security.

WebHook data sample

```
{
  "buildNumber": "12",
  "jobName": "1600Freestyle",
  "jobUrl": "http://localhost:8080/job/1600Freestyle/",
  "buildStatus": "Failed",
  "failReason": [
    {
      "imageId": "1708c4cb79fd",
      "severity": {
        "2": {
          "configured": 0,
          "found": 2
        },
        "3": {
          "configured": 0,
          "found": 20
        },
```

```
        "4": {
          "configured": 0,
          "found": 4
        },
        "5": {
          "configured": 0,
          "found": 3
        }
      },
      "qid": {
        "configured": "256510-256570,176983",
        "found": "176983"
      },
      "cve": {
        "configured": "CVE-2019-3846,CVE-2019-5489,CVE-2019-9500,CVE-2019-
9503,CVE-2019-10126,CVE-2019-11477,CVE-2019-11478,CVE-2019-11479,CVE-2019-
11486,CVE-2019-11599,CVE-2019-11815,CVE-2019-11833,CVE-2019-11884",
        "found": "CVE-2019-3846,CVE-2019-5489,CVE-2019-9500,CVE-2019-9503,CVE-
2019-10126,CVE-2019-11477,CVE-2019-11478,CVE-2019-11479,CVE-2019-11486,CVE-2019-
11599,CVE-2019-11815,CVE-2019-11833,CVE-2019-11884"
      },
      "cvss": {
        "configured": 0,
        "found": 29,
        "version": 2
      },
      "software": {
        "configured": "python-urlgrabber, python3",
        "found": "python3=3.5.3-1"
      }
    }
  ],
  "images": [
    {
      "imageId": "1708c4cb79fd",
      "uuid": "28bb54a0-1415-371d-beb1-8affd626af7a",
      "sha": "1708c4cb79fd6cb3186ccb0523a36ad7902ca2a8c3ddcda8c9ac15c48e9de7b5",
      "size": 1242755393,
      "repo": [
        {
          "registry": "docker.io",
          "tag": "latest",
          "repository": "dock_jenkins_3"
        }
      ],
      "operatingSystem": "Debian Linux 9.5",
      "layersCount": 62,
      "dockerVersion": "1.13.1",
```

```
      "architecture": "amd64",
      "vulnerabilities": {
        "totalVulnerabilities": 29,
        "typeDetected": {
          "Confirmed": 24,
          "Potential": 5
        },
        "severity": {
          "Potential": {
            "1": 0,
            "2": 1,
            "3": 2,
            "4": 1,
            "5": 1
          },
          "Confirmed": {
            "1": 0,
            "2": 1,
            "3": 18,
            "4": 3,
            "5": 2
          }
        },
        "patchable": {
          "yes": 26,
          "no": 3
        }
      }
    }
  ]
}
```

Configuration Details

Provide the following configuration details:

(1) API login information (Select Use Proxy Settings to provide proxy information).

(2) Click Test Connection to verify that the plugin can call the Qualys Container Security API.

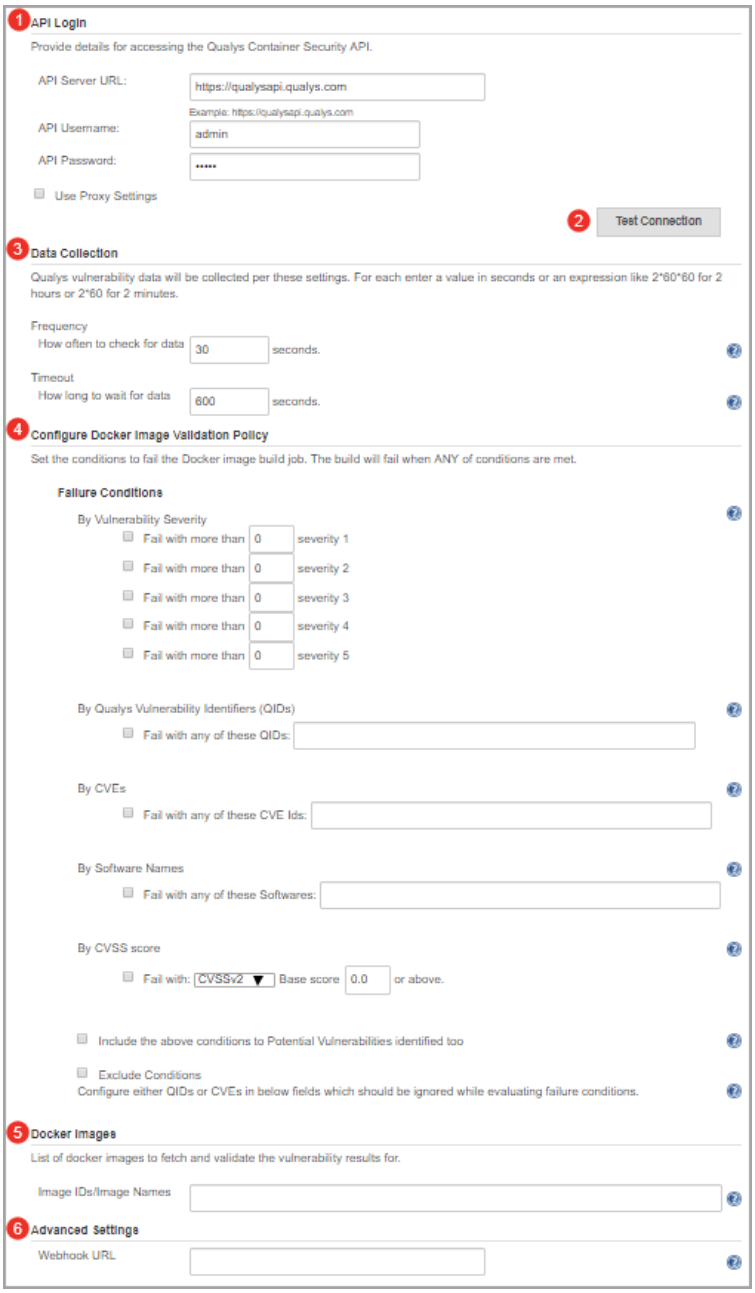
(3) data collection frequency.

(4) build fail conditions.

(5) docker image IDs / image names to check for vulnerabilities.

(6) forward Jenkins job results to a WebHook URL.

When you're ready, click Generate Pipeline Script to



If you are setting a global configuration, you can select a user from the Credential Store to authenticate to the API Server. In case of Job specific configuration, you can provide the credentials in the pipeline / freestyle script.

Note: Use global configuration for scanning images in CI/CD pipeline. See Scanning CI/CD images.

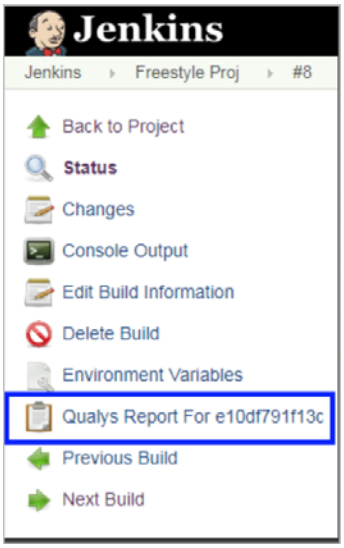
What API Server URL to use?

Qualys maintains multiple platforms. The Qualys platform URL that you should use for API requests depends on the platform where your account is located.

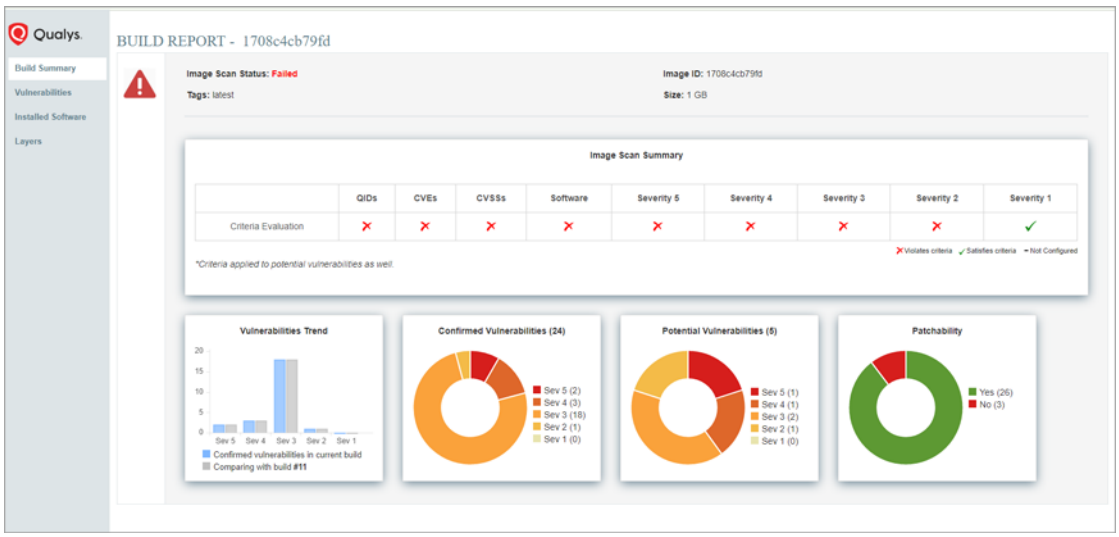
Account Location	Platform URL
Qualys US Platform 1	https://qualysapi.qualys.com
Qualys US Platform 2	https://qualysapi.qg2.apps.qualys.com
Qualys US Platform 3	https://qualysapi.qg3.apps.qualys.com
Qualys EU Platform 1	https://qualysapi.qualys.eu
Qualys EU Platform 2	https://qualysapi.qg2.apps.qualys.eu
Qualys India Platform 1	https://qualysguard.qg1.apps.qualys.in

View Your Qualys Report

In either case - pipeline project or freestyle project - the plugin will generate one report for each docker image in the build. In other words, multiple image Ids will result in multiple report links.



Click any report link to see vulnerability details for the docker image.



The Build Summary provides a dashboard view of your security posture. Go to Vulnerabilities for a list of detected QIDs, Installed Software to see software detected on the docker image, and Layers to view a list of layers the image is made of.

Debugging and Troubleshooting

Where are the logs?

Windows:

Jenkins logs are located on Windows at JENKINS_HOME path.

For example,

Master logs - C:\Program Files (x86)\Jenkins\jenkins.err

Slave logs - C:\Program Files (x86)\Jenkins\logs\slaves\<slave_name>\slave_log_file

Linux:

Master logs - /var/log/jenkins/jenkins.log

Slave logs - /var/jenkins/remoting/logs/remoting.log.0

HTTP codes in API response

All API calls and their responses are logged by the plugin and are visible in the Console Output. Here are the HTTP response codes you may see during plugin execution.

Code	Error	Description
204	No content	Qualys sensor is processing data. You'll see 200 OK when complete.
200	OK	You would see this code in two situations: 1) You might have received partial data from Qualys where image details are available but vulnerability data is not available. 2) Vulnerability data is also available. This is usually the last call, after which the thread for that image Id stops.
500	Internal server error	Qualys service is down or there was an issue processing data.
400	Bad request	Qualys API server is unable to understand the request.
401	Unauthorized	The credentials used for Qualys API server are incorrect or the user does not have access to the APIs.

If you don't see any API calls being made...

Make sure you're correctly passing image Ids to the plugin. When the plugin starts the execution, it will print the image Ids provided and you can see this in the Console Output. Check that the docker image Ids you provided are printed.

Plugin times out, no report seen

The plugin is designed to keep polling the Qualys API until the configured timeout period is reached. If it does not get vulnerability data from Qualys within this period, it stops. In this case, the plugin will fail the build only if you have set any fail-on conditions. Otherwise, it will not fail the build. You will not see any report links since the plugin could not get vulnerability data, and could not prepare a report.

How to fix this?

On the Qualys Cloud Platform, go to Container Security > Assets > Images and verify if the image for which you are checking the vulnerabilities is present in the Images list.

If the image is not present console logs have the following entry:

```
Get scan result API for image e0111ddfea06 returned code : 404;
HTTP Code: 404. Image: Not known to Qualys. Vulnerabilities: To be
processed.. API Response : {"errorCode":"CMS-2002","message":"Data not
available for given Image Id.","timestamp":1554568122039}
```

Ensure that the Qualys Container Sensor is installed on the host where image is being built.

If the image is present console logs have the following entry:

```
Get scan result API for image cef4ca723229 returned code : 200;
Waiting for vulnerabilities data from Qualys for image id cef4ca723229
HTTP Code: 200. Image: known to Qualys. Vulnerabilities: To be processed.
```

Wait for the vulnerabilities data to be uploaded to the Qualys Cloud Platform.

No space left on device

Console logs have the following entry:

```
Apr 04, 2019 11:20:47 AM
hudson.remoting.JarCacheSupport$DownloadRunnable run
WARNING: Failed to resolve a jar 7cab007bbd804ea3a998084524ce236e
java.io.IOException: Failed to write to
/home/dockerAdmin/.jenkins/cache/jars/7C/AB007BBD804EA3A998084524CE236E.
jar
    at
hudson.remoting.FileSystemJarCache.retrieve(FileSystemJarCache.java:150)
    at
hudson.remoting.JarCacheSupport$DownloadRunnable.run(JarCacheSupport.jav
a:110)
    at
java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
    at java.util.concurrent.FutureTask.run(FutureTask.java:266)
    at
```

```
hudson.remoting.AtmostOneThreadExecutor$Worker.run(AtmostOneThreadExecut
or.java:112)
    at java.lang.Thread.run(Thread.java:748)
Caused by: java.nio.file.FileSystemException:
/home/dockerAdmin/.jenkins/cache/jars/7C: No space left on device
```

Possible reasons could be:

1) Slave machine is running out of disk space.

Use the command **`df -h`** to check the disk space. Observe 'Use%' and 'Available' space in the output of the command.

Ensure that you have provisioned enough disk space on the host. You may remove unnecessary files to increase free disk space.

2) iNodes are exhausted during the Jenkins plugin job.

Use the command **`df -i`** to check the iNodes. Observe 'IFree'count and 'IUse%' in the output of the command.

If IUse% is at 100 or near, then large number of small (cached) files are present on the host.

To delete the cached files from the host:

- Run the following command to list all the directories and files:

```
for i in /*; do echo $i; find $i | wc -l; done
```

- Locate the directory with unusually large number of small files, and the run the following command to see where exactly the files are:

For example, if the directory is /home:

```
for i in /home/*; do echo $i; find $i |wc -l; done
```

- Run the following command to remove those files:

```
sudo rm -rf <path of the directory with small files>
```

Incorrect checksum of retrieved jar

Console logs have the following entry:

```
WARNING: Failed to resolve a jar 4fd2a7e0ee23f360d678b2af7903dab0
java.io.IOException: Failed to write to
/home/dockerAdmin/.jenkins/cache/jars/4F/D2A7E0EE23F360D678B2AF7903DAB0.
jar
    at
hudson.remoting.FileSystemJarCache.retrieve(FileSystemJarCache.java:150)
    at
hudson.remoting.JarCacheSupport$DownloadRunnable.run(JarCacheSupport.jav
```

```
a:110)
    at
java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
    at java.util.concurrent.FutureTask.run(FutureTask.java:266)
    at
hudson.remoting.AtmostOneThreadExecutor$Worker.run(AtmostOneThreadExecut
or.java:112)
    at java.lang.Thread.run(Thread.java:748)
Caused by: java.io.IOException: Incorrect checksum of retrieved jar:
```

Jenkins writes the jar required for job execution in its cache on the slave host. This error comes in case the already existing jar in cache is corrupted.

To fix this, clear the cache. The logs show the location of the cache.

For example, in the above logs the cache is located at:

`/home/dockerAdmin/.jenkins/cache/jars/4F/D2A7E0EE23F360D678B2AF7903DAB0.jar`

Execute the command **`-ll`** to verify that Jenkins has permission to write to the folder:

`/home/dockerAdmin/.jenkins/`

Ensure that the cache directory `/home/dockerAdmin/.jenkins/cache/` is writable to the user with which the slave is added to the Jenkins Server. To give permissions, add the respective user owner group OR change the owner to the user with which the Jenkins slave is configured.

Want to contact Support?

Access online support information at www.qualys.com/support/

You'll typically need to provide the following information for Qualys Jenkins plugin issues:

- Jenkins version

- Java version on which Jenkins is running

- Version of the Qualys Vulnerability Analysis Plugin

- Jenkins master-slave topology

- Whether the Docker daemon is on Jenkins master or Jenkins slave or remote host