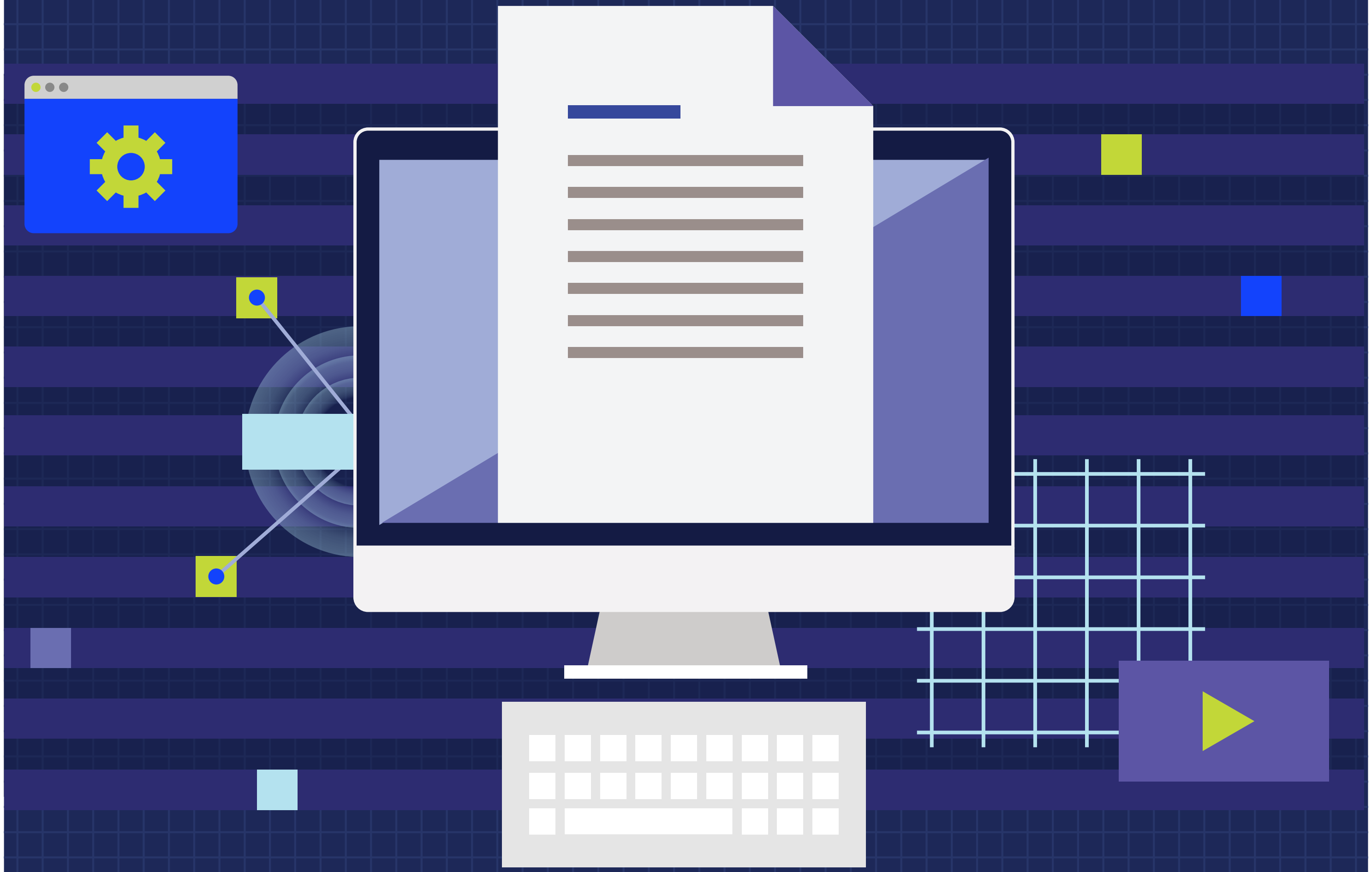
Twistlock

Enterprise Security. DevOps Agility.



GUIDE TO PCI COMPLIANCE FOR CONTAINERS

TABLE OF CONTENTS

EXECUTIVE SUMMARY

If you handle credit card data, it is your responsibility to secure it. Your customers trust you not only to deliver the products they purchase, but to keep their data safe.

The Payment Card Industry (PCI) Data Security Standard (DSS) defines a set of guidelines for securing credit card data in order to minimize theft, fraud, and misuse. Anyone that accepts, transmits, or stores cardholder data must comply to the PCI DSS. This includes merchants, who accept credit cards, and service providers, such as payment gateways, who transmit cardholder data.

With the advent of microservices, it is likely that a growing part of your platform is built on containers. One of the early steps in preparing for a PCI DSS assessment is identifying the system components included in, or connected to, the cardholder data environment (CDE). These in-scope components are subject to the PCI DSS requirements. They can include network devices, servers, hypervisors, and application software, such as a directory service. If you have containers that been determined to be in-scope, then Twistlock can be deployed to help bring them into compliance with the PCI DSS.

Examples of in-scope system components include a containerized nodejs app that serves an ecommerce website, a containerized MongoDB database that stores cardholder data to streamline checkout for current customers, or a containerized OpenLDAP service that authenticates access to hosts and applications in the CDE.

This guide is written for members of both the technical and compliance teams that work with containers in the CDE. It shows how Twistlock's capabilities can help realize and maintain ongoing compliance with the PCI DSS. It is most applicable to service providers and large merchants, which initiate a large number of transactions, process payments, and/or store cardholder data to handle recurring payments and one-click checkout. These types of companies bear the burden of showing their banks how they meet each requirement of the PCI DSS in a yearly audit. Smaller merchants often limit their scope by outsourcing payment processing to PCI DSS compliant service providers. In this way, small merchants might only be responsible for submitting a simple self-assessment questionnaire (SAQ), while the burden of a full audit falls to the service provider.

INTRODUCTION

This guide shows you how to deploy and configure Twistlock to meet specific requirements in the Payment Card Industry (PCI) Data Security Standard (DSS) for containers in your cardholder data environment (CDE).

For each relevant requirement in the PCI DSS, the guide describes the applicable Twistlock feature or technology, shows how it can be configured to satisfy the requirement, and provides links to the official documentation for more detailed information.

This guide has three major sections:

- Bringing the Twistlock package into compliance with the PCI DSS
  Addresses requirements from section 2 of the PCI DSS.
- Configuring Twistlock to bring your container environment into compliance with the PCI DSS
  Addresses requirements from sections 1, 5, 6, 7, 8, and 10 of the PCI DSS.
- Sample policy file
  Shows you how to use the Twistlock API to deploy the policies recom mended in this guide.

NOTE: In order to access links to the official documentation, you must have a Twistlock subscription. A subscription to the Developer Edition is free. For more information, and to sign up for Developer Edition, go to www.twistlock.com.

TWISTLOCK

Twistlock is a security suite that is purpose-built for containers. Twistlock lets you centrally declare security policies, and then enforces your policies across your container environment. In general, a container environment includes all your images, containers, hosts, registries, and Docker daemons.
With Twistlock, your data never leaves your jurisdiction. All scanning, analysis, and detection is executed inside your environment only.

PAYMENT CARD INDUSTRY (PCI) DATA SECURITY STANDARD (DSS)

The PCI DSS is a set of requirements designed to protect cardholder data and reduce credit card fraud. It is an industry standard defined by the PCI Security Standards Council. The Council was formed by the major card brands (Visa, MasterCard, American Express, Discover, and JCB) to develop a single standard that is recognized by all card brands.

The PCI DSS applies to any company that handles (accepts, transmits, or stores) cardholder data. If customers pay you with a credit card, or if you handle any part of the payment transaction as a service to another merchant, then the requirements in the PCI DSS apply to you.

The PCI DSS describes a baseline for how to secure system components in the cardholder data environment (CDE). System components is a PCI DSS term that encompasses routers, switches, hypervisors, and application software. It boils down to the systems that handle sensitive cardholder data, at rest and in motion, the systems segregate the CDE, and the systems that control access to the CDE. For this guide, we're interested in the application software in the CDE and, in particular, the portions that have been deployed as containers.

SCOPE

This guide identifies the requirements from the PCI DSS that Twistlock can be configured to satisfy. It is written against version 3.2 of the standard.

This guide is not a comprehensive treatment of the PCI DSS. It will help you in your effort to achieve PCI compliance, but it will not get you fully compliant. As such, not all requirements in the PCI DSS are addressed. In some cases, the PCI DSS requirements are not applicable to containerized workloads (i.e. Requirement 9: Restrict physical access to cardholder data). In other cases, it's not clear how the requirements should be addressed without specific details about your environment.

BRINGING THE TWISTLOCK PACKAGE INTO COMPLIANCE WITH THE PCI DSS

The default installation of Twistlock is reasonably secure. However, there are some settings designed to optimize the out-of-the-box experience that conflict with PCI DSS requirements.

The following table summarizes the PCI DSS requirements addressed in this section:

PCI DSS	Summary of requirement
Requirement 2.1	Remove unnecessary default accounts.
Requirement 2.2	Develop configuration standards for all system components.
Requirement 2.2.1	Implement only one primary function per server.
Requirement 2.2.3	Implement additional security features.
Requirement 2.3	Encrypt all non-console administrative access using strong cryptography.

REQUIREMENT 2.1

Always change vendor-supplied defaults and remove or disable unnecessary default accounts before installing a system on the network.

TWISTLOCK CONFIGURATION
The default Twistlock installation comes with a default user with administrator-level privileges (admin/admin).

To ensure that Twistlock never appears on the network with this default account, specify the admin username and password in ***twistlock.cfg*** before installing Twistlock:

1 | Open ***twistlock.cfg*** for editing.

2 | Set values for the following parameters:

```
ADMIN_USERNAME=
ADMIN_PASSWORD=
```

3 | Install Twistlock using the normal procedure.

4 | Delete ***twistlock.cfg*** so that the admin user account credentials cannot be compromised.

NOTE: A copy of your ***twistlock.cfg*** file is kept in /var/lib/twistlock/scripts for future reference. Because this file contains sensitive information, it is secured with the following permissions:

- Ownership and group ownership for the /var/lib/twistlock directory is set to root, and the permissions are set to 750 (writable by owner, readable by owner and group).
- Ownership and group ownership for ***twistlock.cfg*** is set to root, and permissions are set to 640 (writable by owner, readable by group).

MORE INFORMATION
Install Twistlock

REQUIREMENT 2.2

Develop configuration standards for all system components. Assure that these standards address all known security vulnerabilities and are consistent with industry-accepted system hardening standards.

HOW DOES TWISTLOCK HELP
Twistlock lets you build out policies based on the best practices defined in the Center for Internet Security (CIS) Docker Benchmark. The CIS Docker Benchmark is a guide for establishing a secure configuration posture for Docker. It recommends how to:

- Configure host machines.
- Configure Docker daemons.
- Secure Docker configuration files.
- Build container images.
- Run containers.

For each recommendation the CIS Docker Benchmark, Twistlock has implemented a corresponding compliance check.

TWISTLOCK CONFIGURATION
No action required. Twistlock ships with a default rule that raises an alert when a compliance check from the CIS Docker Benchmark fails. You can review the compliance checks in Console under Defend > TRUST > COMPLIANCE.

MORE INFORMATION
Compliance screencast
Managing compliance

REQUIREMENT 2.2.1

Implement only one primary function per server to prevent functions that require different security levels from co-existing on the same server.

TWISTLOCK CONFIGURATION

Twistlock Console should be installed on a dedicated host. Install Twistlock using the onebox target, which installs the following components:

- **Console**: Management interface.
- **Defender**: Registry scanner.

MORE INFORMATION

System requirements
Install Twistlock

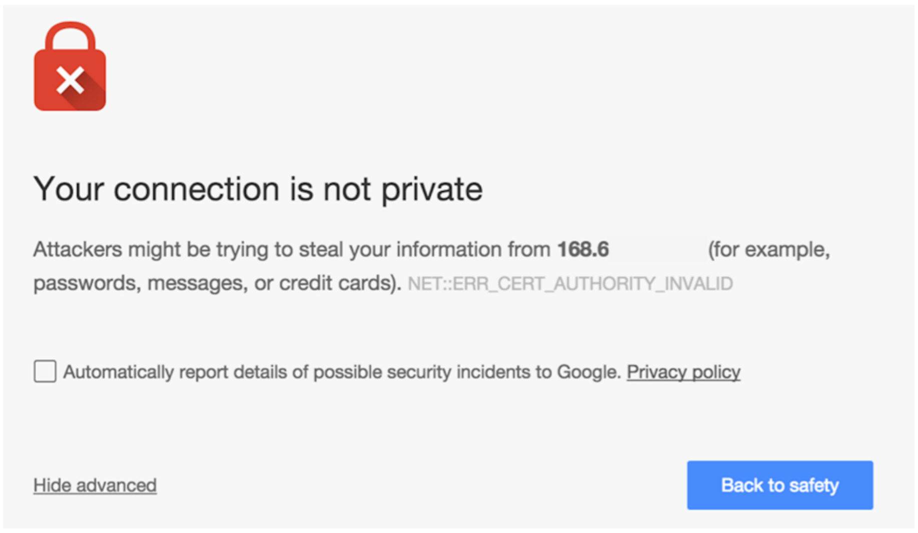
REQUIREMENT 2.2.3

Implement additional security features for any required services, protocols, or daemons that are considered to be insecure.

TWISTLOCK CONFIGURATION

By default, Twistlock secures access to Console's web portal and API with a self-signed certificate When you access Console's web portal with this setup, the browser flags the portal as untrusted with a warning message.

For example, the Chrome browser shows the following message when you access Console over HTTPS with Twistlock's self-signed certificate:



Your connection is not private

Attackers might be trying to steal your information from **168.6** (for example, passwords, messages, or credit cards). NET::ERR_CERT_AUTHORITY_INVALID

☐ Automatically report details of possible security incidents to Google. Privacy policy

You can resolve these warnings by installing your own certificate that proves your server's identity to the client.

MORE INFORMATION

[Custom certs for Console access]

REQUIREMENT 2.3

Encrypt all non-console administrative access using strong cryptography.

TWISTLOCK CONFIGURATION
For convenience, the default Twistlock installation lets you connect to Console (web portal and API) over HTTP on port 8081. HTTP is a clear-text, unencrypted protocol. Eavesdroppers can easily intercept sensitive data transmitted over HTTP. You should disable the HTTP management port, and force users to connect to Console over HTTPS on port 8083.

To disable the HTTP management port, edit ***twistlock.cfg*** before installing Twistlock:

1 | Open ***twistlock.cfg*** for editing.

2 | Disable the HTTP management port by leaving the value empty:

```
MANAGEMENT_PORT_HTTP=
```

3 | Install Twistlock using the normal procedure.

NOTE: If you have already installed Twistlock, and you want to disable the HTTP management port, re-run the installer with an updated configuration file.

MORE INFORMATION
Install Twistlock
Reconfigure Twistlock

CONFIGURING TWISTLOCK TO BRING YOUR CONTAINER ENVIRONMENT INTO COMPLIANCE WITH THE PCI DSS

This section shows you how to configure Twistlock and install security policies to bring your container environment into compliance with the PCI DSS, where the container environment consists of any host, Docker daemon, image, or container in the cardholder data environment.

This section addresses requirements from sections 1, 5, 6, 7, 8, and 10 of the PCI DSS.

The following table summarizes the PCI DSS requirements addressed in this section. The Policy file column indicates that the requirement can be satisfied with a Twistlock compliance rule. All recommended compliance rules are supplied in a sample policy file that can be installed with the Twistlock API.

PCI DSS	SUMMARY OF REQUIREMENT	POLICY FILE
Requirement 1.2	Build firewall and router configurations that restrict connections between untrusted networks and the cardholder data environment.	Yes
Requirement 1.2.1	Restrict unnecessary inbound and outbound traffic to/from the cardholder data environment.	Yes
Requirement 5.1	Deploy anti-virus software on all systems commonly affected by malicious software.	
Requirement 5.2	Ensure that all anti-virus mechanisms are maintained	
Requirement 5.3	Ensure that anti-virus mechanisms are actively running and cannot be disabled or altered by users.	
Requirement 6.1	Establish a process to identify security vulnerabilities.	
Requirement 6.4.1	Separate development/test environments from production environments with access controls.	
Requirement 7.1	Limit access to system components and cardholder data to only those individuals whose job requires such access.	
Requirement 7.1.2	Restrict access to privileged user IDs to least privileges necessary to perform job responsibilities.	Yes
Requirement 7.2.3	Default "deny-all" setting for your access control system(s).	
Requirement 8.1.1	Assign all users a unique ID before allowing them to access system components or cardholder data.	
Requirement 8.2	Besides unique IDs, ensure proper user-authentication management for non-consumer users and administrators on all system components.	
Requirement 8.5	Do not use group, shared, or generic IDs or passwords.	
Requirement 10.1	Implement audit trails to link all access to system components to each individual user.	
Requirement 10.3	Audit trail entries must record some minimum information.	
Requirement 10.5.1	Limit viewing of audit trails to those with a job-related need.	
Requirement 10.5.2	Protect audit trail files from unauthorized modifications.	
Requirement 10.5.3	Promptly back up audit trail files to a centralized log server or media that is difficult to alter.	

REQUIREMENT 1.2

Build firewall and router configurations that restrict connections between untrusted networks and any system components in the cardholder data environment.

DISCUSSION
Every container should have its own network namespace. Network namespaces enforce isolation between containers, and between containers and your host. Containers can be configured to share the host's network stack, this setup is insecure because the container would have full access to the host's network interfaces.

TWISTLOCK CONFIGURATION GUIDE
Create a compliance rule that raises an alert when a container is started with the --net=host option.

You can either create this compliance rule manually, or you can import this rule, along with all other recommended rules from this guide, using the Twistlock API.

- To install this rule manually, see Managing compliance.
  Select compliance check ID 59 (CIS 5.9).
- To install this rule with the Twistlock API, see Sample policy file.

MORE INFORMATION
Managing compliance policies with the API

REQUIREMENT 1.2.1

Restrict inbound and outbound traffic to that which is necessary for the cardholder data environment, and specifically deny all other traffic.

DISCUSSION
Network traffic between containers on a host is unrestricted by default. A breached container could intercept packets on the default container network and steal sensitive information. For containers that must communicate with each other, you should explicitly link them together. All other communication between containers should be blocked.

TWISTLOCK CONFIGURATION GUIDE
Create a compliance rule that checks that the Docker daemon on your host is started with the --icc=-false option. This option disables inter-container communication unless explicitly specified.

When you want two containers to be able to communicate, start the container (docker run) with the --link= option, Docker inserts a pair of iptables ACCEPT rules so that the new container can connect to the ports exposed by the other container.

You can either create this compliance rule manually, or you can import this rule, along with all other recommended rules from this guide, using the Twistlock API.

- To install this rule manually, see Managing compliance.
  Select compliance check ID 21 (CIS 2.1).
- To install this rule with the Twistlock API, see Sample policy file.

MORE INFORMATION
Managing compliance policies with the API

REQUIREMENT 5.1

Deploy anti-virus software on all systems commonly affected by malicious software.

DISCUSSION
Twistlock runtime defense monitors running containers for malware being downloaded into the container file system and connections to banned IP addresses.

TWISTLOCK CONFIGURATION GUIDE
Install Defender on every host that runs containers. Twistlock runtime defense is enabled by default.

MORE INFORMATION
Install Defender
Runtime defense screencast

REQUIREMENT 5.2

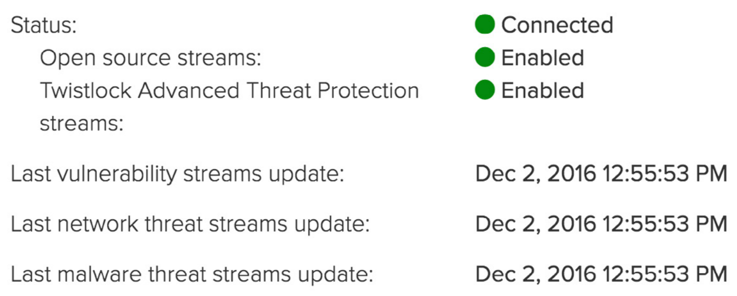
Ensure that all anti-virus mechanisms are maintained as follows:

- Are kept current
- Perform periodic scans.
- Generate audit logs which are retained per PCI DSS Requirement 10.7.

DISCUSSION
The following subsections describe how Twistlock helps meet the three criteria in this requirement.

Anti-virus mechanisms are kept current
The Twistlock Intelligence Stream delivers threat data to Console. When Twistlock updates the feed, your installation is automatically updated with the latest threat data.

Console shows the status of the Twistlock Intelligence Stream connection, along with a timestamp for the latest update. To get to the status page, open Console, and then go to
Configure > SYSTEM > INTELLIGENCE.



Status:	● Connected
Open source streams:	● Enabled
Twistlock Advanced Threat Protection streams:	● Enabled
Last vulnerability streams update:	Dec 2, 2016 12:55:53 PM
Last network threat streams update:	Dec 2, 2016 12:55:53 PM
Last malware threat streams update:	Dec 2, 2016 12:55:53 PM

Perform periodic scans
Rather than periodic scans, Defender continually monitors the container file system for changes. When a file is written to a container, Defender immediately analyzes it for malware.

Audit logs
When Twistlock detects malware in a container, a file system audit event is emitted. Console aggregates audit events from all containers across your environment.

By itself, Twistlock cannot fulfill PCI DSS Requirement 10.7. However, you can configure Twistlock to send all audit events to syslog in RFC5424-compliant format. If you have a centralized syslog collector that is backed up according to PCI DSS Requirement 10.7, you can easily integrate Twistlock into your existing infrastructure to meet this requirement.

TWISTLOCK CONFIGURATION GUIDE
By default, Twistlock keeps threat data up to date and monitors containers for malware. Assuming your central syslog collector meets PCI DSS requirement 10.7, integrate Twistlock with syslog.

MORE INFORMATION
Syslog integration

REQUIREMENT 5.3

Ensure that anti-virus mechanisms are actively running and cannot be disabled or altered by users, unless specifically authorized by management on a case-by-case basis for a limited time period.

DISCUSSION
Twistlock ships with default access controls that prevent Console or Defender from being disabled.

- A default deny-all rule blocks users from disabling the Twistlock containers with Docker Engine commands. The default deny-all rule prevents any user from running any Docker command on any container in your environment. This means that no one can run docker stop, docker rm, or docker rmi on the Twistlock containers.
- The Console web management portal only allows users that have Administrator, Operator, Defender Manager, or Auditor roles to decommission Defenders.

Besides access control policies that block Twistlock from being disabled, you can also set up email alerts on Defender health events, which notify you when Defenders are decommissioned or they disconnect from Console.

TWISTLOCK CONFIGURATION GUIDE
Set up email alerts for Defender health events.

MORE INFORMATION
Email alerts
User roles

REQUIREMENT 6.1

Establish a process to identify security vulnerabilities, using reputable outside sources for security vulnerability information, and assign a risk ranking (for example, as "high," "medium," or "low") to newly discovered security vulnerabilities.

By default, Twistlock periodically scans all images and containers in your environment for CVEs and zero-day vulnerabilities. The default scan interval is 24 hours, and the size of this interval is configurable. Scans are also immediately triggered when images or containers first appear on a host (for example, after you run docker pull). Alerts are raised for vulnerabilities with a CVSS score of 4.0 or higher, and this threshold is configurable. Scan reports are published in Console. Every reported vulnerability comes with a link to the information source.

The following screenshot shows a scan report:

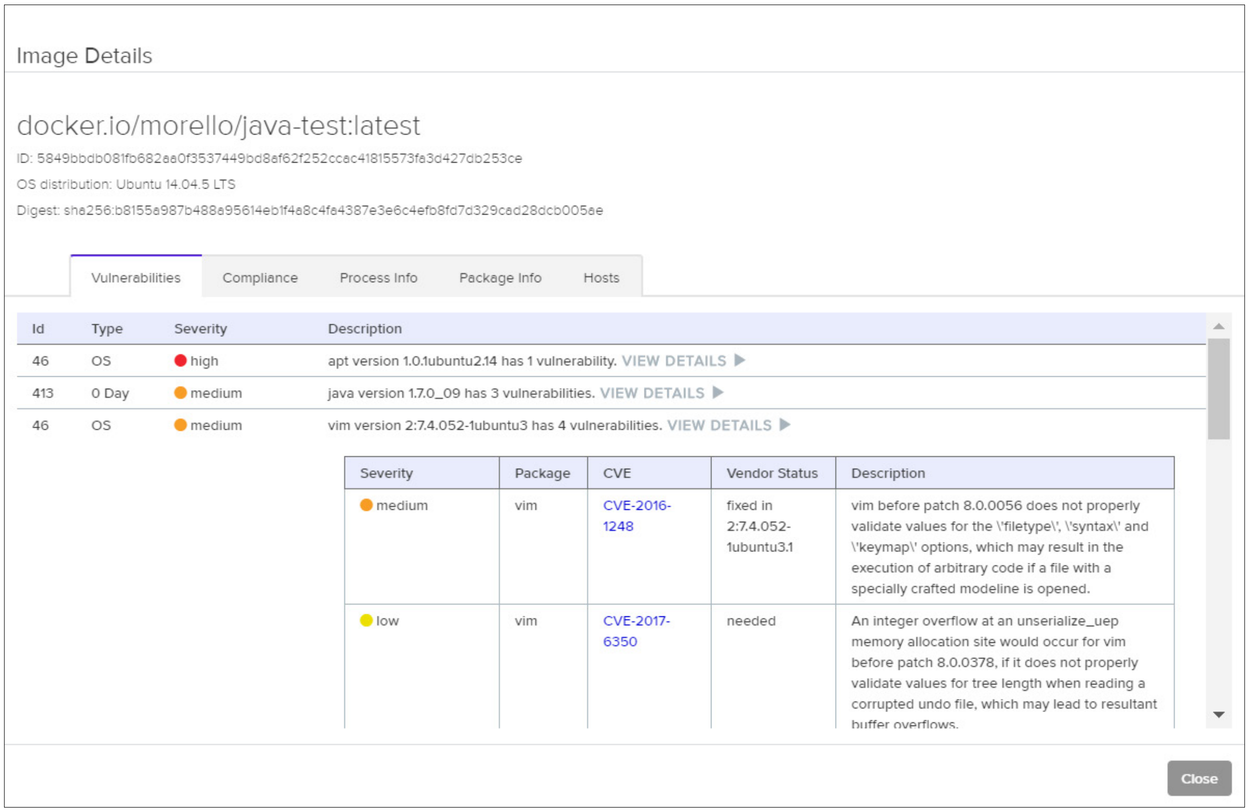


Image Details

docker.io/morello/java-test:latest

ID: 5849bbdb081fb682aa0f3537449bd8af62f252ccac41815573fa3d427db253ce

OS distribution: Ubuntu 14.04.5 LTS

Digest: sha256:b8155a987b488a95614eb1f4a8c4fa4387e3e6c4efb8fd7d329cad28dcb005ae

Id	Type	Severity	Description
46	OS	● high	apt version 1.0.1ubuntu2.14 has 1 vulnerability. VIEW DETAILS ▶
413	0 Day	● medium	java version 1.7.0_09 has 3 vulnerabilities. VIEW DETAILS ▶
46	OS	● medium	vim version 2:7.4.052-1ubuntu3 has 4 vulnerabilities. VIEW DETAILS ▶

Severity	Package	CVE	Vendor Status	Description
● medium	vim	CVE-2016-1248	fixed in 2:7.4.052-1ubuntu3.1	vim before patch 8.0.0056 does not properly validate values for the \'filetype\', \'syntax\' and \'keymap\' options, which may result in the execution of arbitrary code if a file with a specially crafted modeline is opened.
● low	vim	CVE-2017-6350	needed	An integer overflow at an unserialize_uep memory allocation site would occur for vim before patch 8.0.0378, if it does not properly validate values for tree length when reading a corrupted undo file, which may lead to resultant buffer overflows.

Vulnerabilities in the scan report are assigned a severity of high, medium, or low. Severity is based on the Common Vulnerability Scoring System (CVSS) score assigned to the CVE. When possible, Twistlock uses the vendor's score in order to capture the severity accurately.

- **High:** The vulnerability has a CVSS base score that ranges from 8.0 to 10.0.
- **Medium:** The vulnerability has a CVSS base score that ranges from 4.0-7.9.
- **Low:** The vulnerability has a CVSS base score that ranges from 0.0-3.9.

Twistlock vulnerability scanning can be integrated into other processes in your organization:

- Jenkins and TeamCity plugins let you incorporate vulnerability scanning into your continuous integration (CI) pipeline so that developers can identify and fix security defects before images get to the registry.
- Registry scanning lets you identify critical vulnerabilities before images go into production.
- TwistScan, a stand-alone program, lets you build vulnerability scanning into custom tooling.

TWISTLOCK CONFIGURATION GUIDE
Install Defender on any host that runs containers. Defender automatically scans all images and containers on the host for vulnerabilities.

MORE INFORMATION
Jenkins plugin
TeamCity plugin
Configure registry scans
TwistScan

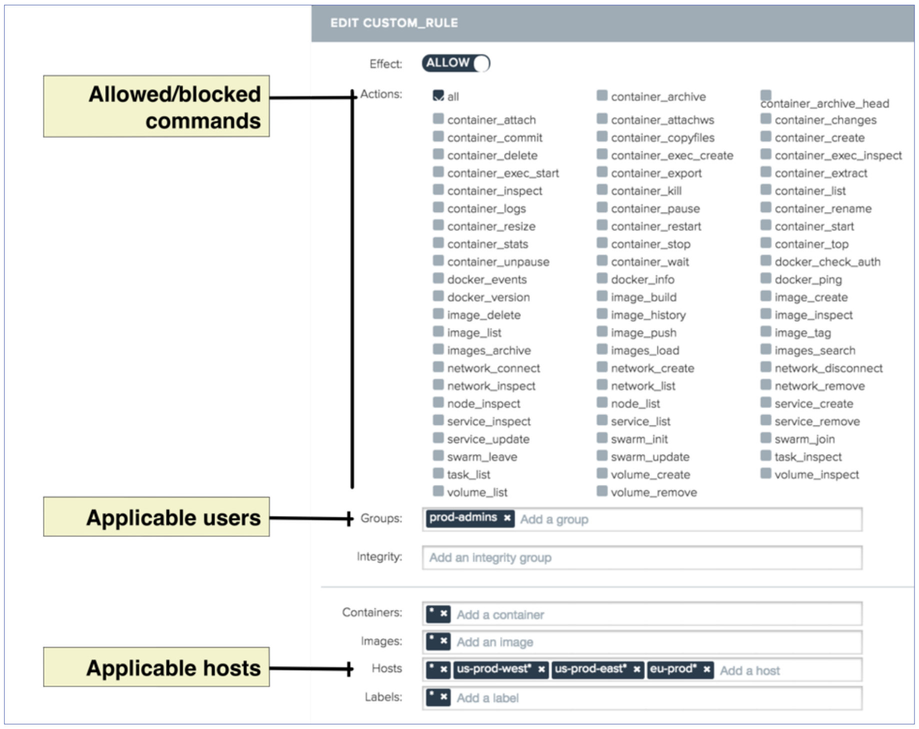
REQUIREMENT 6.4.1

Separate development/test environments from production environments, and enforce the separation with access controls.

DISCUSSION

Twistlock access control rules specify what users can take what actions against what resources. If you have established a standardized naming scheme for the hosts in your environment, you can separate and target access control rules for test, development, and production environments.

The following screenshot shows the dialog for creating an access control rule for Docker Engine commands. Twistlock also supports access control rules for Docker Swarm and Kubernetes commands.



EDIT CUSTOM_RULE

Effect: ALLOW

Actions:

✓ all	container_archive	container_archive_head
container_attach	container_attachws	container_changes
container_commit	container_copyfiles	container_create
container_delete	container_exec_create	container_exec_inspect
container_exec_start	container_export	container_extract
container_inspect	container_kill	container_list
container_logs	container_pause	container_rename
container_resize	container_restart	container_start
container_stats	container_stop	container_top
container_unpause	container_wait	docker_check_auth
docker_events	docker_info	docker_ping
docker_version	image_build	image_create
image_delete	image_history	image_inspect
image_list	image_push	image_tag
images_archive	images_load	images_search
network_connect	network_create	network_disconnect
network_inspect	network_list	network_remove
node_inspect	node_list	service_create
service_inspect	service_list	service_remove
service_update	swarm_init	swarm_join
swarm_leave	swarm_update	task_inspect
task_list	volume_create	volume_inspect
volume_list	volume_remove	

Groups: prod-admins Add a group

Integrity: Add an integrity group

Containers: * Add a container

Images: * Add an image

Hosts: * us-prod-west* us-prod-east* eu-prod* Add a host

Labels: * Add a label

WHERE

- **Allowed/blocked commands**
  Selects the Docker Engine commands that are allowed to run, or that should be blocked.
- **Applicable users**
  Specifies the groups for which this rule applies. Groups provide an easy way to manage privileges for large numbers of users. In this example, a group called prod-admins contains all users that require privileges to administer the production environment.
- **Applicable hosts**
  Specifies where the commands in this rule can be run. In this example, prod-admins can run all Docker commands on any container or image on hosts named us-prod-west* or us-prod-east*.

For example, a developer named Bruce is a member of the engineering group. You create a rule that grants full access to all commands to users in the engineering group on all hosts in the development environment. Combined with the default deny-all rule, Bruce would be able to do his work in the development environment, but he would be blocked from running any Docker command in any other environment, including the production environment.

TWISTLOCK CONFIGURATION GUIDE

Before you can start creating access policies, every user in the system must have a unique ID.

1 | Integrate Twistlock with your organization's directory service so that you can re-use the identities and groups already set up in your directory service. Twistlock sup ports Active Directory, OpenLDAP, and SAML.

If you do not have a directory service, Twistlock provides a mechanism to create local users and groups.

2 | Create the rules required to realize your access control policy. By default, Twistlock blocks all actions by all users on all resources. You must build out your policies by creating rules that explicitly whitelist approved actions.

MORE INFORMATION

Access control screencast

Integration with directory services:

- Active Directory
- OpenLDAP
- SAML

Access control rules:

- Docker Engine
- Docker Swarm
- Kubernetes

REQUIREMENT 7.1

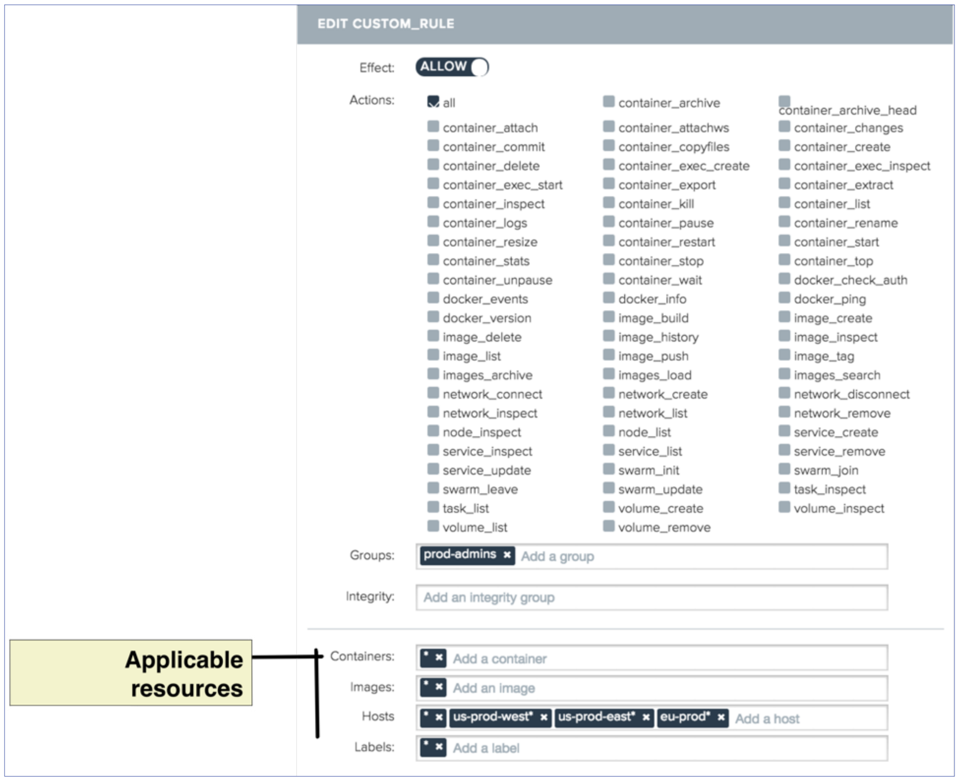
Limit access to system components and cardholder data to only those individuals whose job requires such access.

DISCUSSION
Twistlock lets you limit access to the images and containers in your environment with access control rules. Each access policy rule lets you target:

- Specific users (using groups).
- Specific commands: Docker Engine, Docker Swarm, and Kubernetes commands.
- Specific resources: Containers, images, and hosts.

The response to this requirement is very similar to the response to Requirement 6.4.1. In Requirement 6.4.1, you enforce the separation of test, development, and production environments with access control rules that target specific hosts. In this requirement, you limit access to any resource that holds sensitive cardholder data, which could include not just specific hosts, but also specific images and/or containers. The parameters in the underlying access control rule lets you target any or all of these resources.

The following screenshot shows an access control rule with the resource section high-lighted:



EDIT CUSTOM_RULE

Effect: ALLOW

Actions:

☑ all	☐ container_archive	☐ container_archive_head
☐ container_attach	☐ container_attachws	☐ container_changes
☐ container_commit	☐ container_copyfiles	☐ container_create
☐ container_delete	☐ container_exec_create	☐ container_exec_inspect
☐ container_exec_start	☐ container_export	☐ container_extract
☐ container_inspect	☐ container_kill	☐ container_list
☐ container_logs	☐ container_pause	☐ container_rename
☐ container_resize	☐ container_restart	☐ container_start
☐ container_stats	☐ container_stop	☐ container_top
☐ container_unpause	☐ container_wait	☐ docker_check_auth
☐ docker_events	☐ docker_info	☐ docker_ping
☐ docker_version	☐ image_build	☐ image_create
☐ image_delete	☐ image_history	☐ image_inspect
☐ image_list	☐ image_push	☐ image_tag
☐ images_archive	☐ images_load	☐ images_search
☐ network_connect	☐ network_create	☐ network_disconnect
☐ network_inspect	☐ network_list	☐ network_remove
☐ node_inspect	☐ node_list	☐ service_create
☐ service_inspect	☐ service_list	☐ service_remove
☐ service_update	☐ swarm_init	☐ swarm_join
☐ swarm_leave	☐ swarm_update	☐ task_inspect
☐ task_list	☐ volume_create	☐ volume_inspect
☐ volume_list	☐ volume_remove	

Groups: prod-admins × Add a group

Integrity: Add an integrity group

Containers: * × Add a container

Images: * × Add an image

Hosts * × us-prod-west* × us-prod-east* × eu-prod* × Add a host

Labels: * × Add a label

IMPLEMENTATION GUIDANCE
See Requirement 6.4.1

REQUIREMENT 7.1.2

Restrict access to privileged user IDs to least privileges necessary to perform job responsibilities.

DISCUSSION

The default user in containers is root. In general, you should never run the application in your container as root. Running as root increases the risk of privilege escalation, where a compromised container gains access to resources outside the container with elevated privileges.

TWISTLOCK CONFIGURATION GUIDE

Twistlock provides two compliance checks that verify containers are started as non-root users:

- An image is not created with a user. (CIS 4.1, ID 41).
- A container is running as root (ID 599).

Compliance check 41 validates that an image is configured to run the first process in the container as a non-root user. Developers can specify a default user with the Dockerfile USER instruction. The USER instruction sets the user name or UID to use when running the image.

Compliance check 599 validates the container is actually started as a non-root user. The USER instruction sets the default user for the container, but this build-time setting can be overridden at run-time with the --user option. Compliance check 599 verifies that an operator does not try to override the default USER setting with the --user root option.

You can either create these compliance rules manually, or you can import this rule, along with all other recommended rules from this guide, using the Twistlock API.

- To install this rule manually, see Managing compliance.
  Select compliance check with ID 41 (CIS 4.1) and 599.
- To install this rule with the Twistlock API, see Sample policy file.

MORE INFORMATION

Compliance screencast
Managing compliance policies with the API

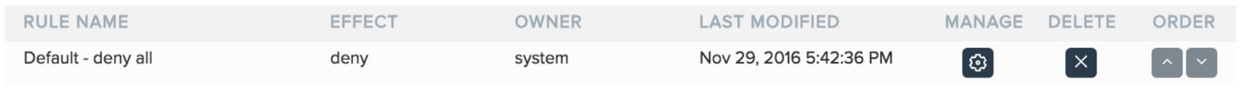
REQUIREMENT 7.2.3

Default “deny-all” setting.

For additional context, see requirement 7.2: Establish an access control system(s) for systems components that restricts access based on a user’s need to know, and is set to “deny all” unless specifically allowed.

DISCUSSION
Twistlock ships with a default policy that blocks all actions by all users (including the default admin user) on all resources.

To see the default deny-all rule, log into Console, then go to Defend > ACCESS > DOCKER. Click on the MANAGE button to see how the rule is configured.



RULE NAME	EFFECT	OWNER	LAST MODIFIED	MANAGE	DELETE	ORDER
Default - deny all	deny	system	Nov 29, 2016 5:42:36 PM			

TWISTLOCK CONFIGURATION GUIDE
No action required. Twistlock ships with a default deny-all rule by default.

REQUIREMENT 8.1.1

Assign all users a unique ID before allowing them to access system components or cardholder data.

DISCUSSION
Container resources protected by Defender can only be accessed by users that have credentials. Credentials uniquely identify a user and enable access policies to be applied.

You can integrate Twistlock with your organization's directory service so that you can re-use identities and credentials. If you do not have a directory service, Twistlock provides a system to create local users.

TWISTLOCK CONFIGURATION GUIDE
Install Defender on all hosts that run containers, then integrate Twistlock with your enterprise directory service. Twistlock supports Active Directory, OpenLDAP, and SAML.

MORE INFORMATION
Integration with a directory service:

- Active Directory
- OpenLDAP
- SAML

REQUIREMENT 8.2

In addition to assigning a unique ID, ensure proper user-authentication management for non-consumer users and administrators on all system components by employing at least one of the following methods to authenticate all users:

- Something you know, such as a password or passphrase
- Something you have, such as a token device or smart card
- Something you are, such as a biometric.

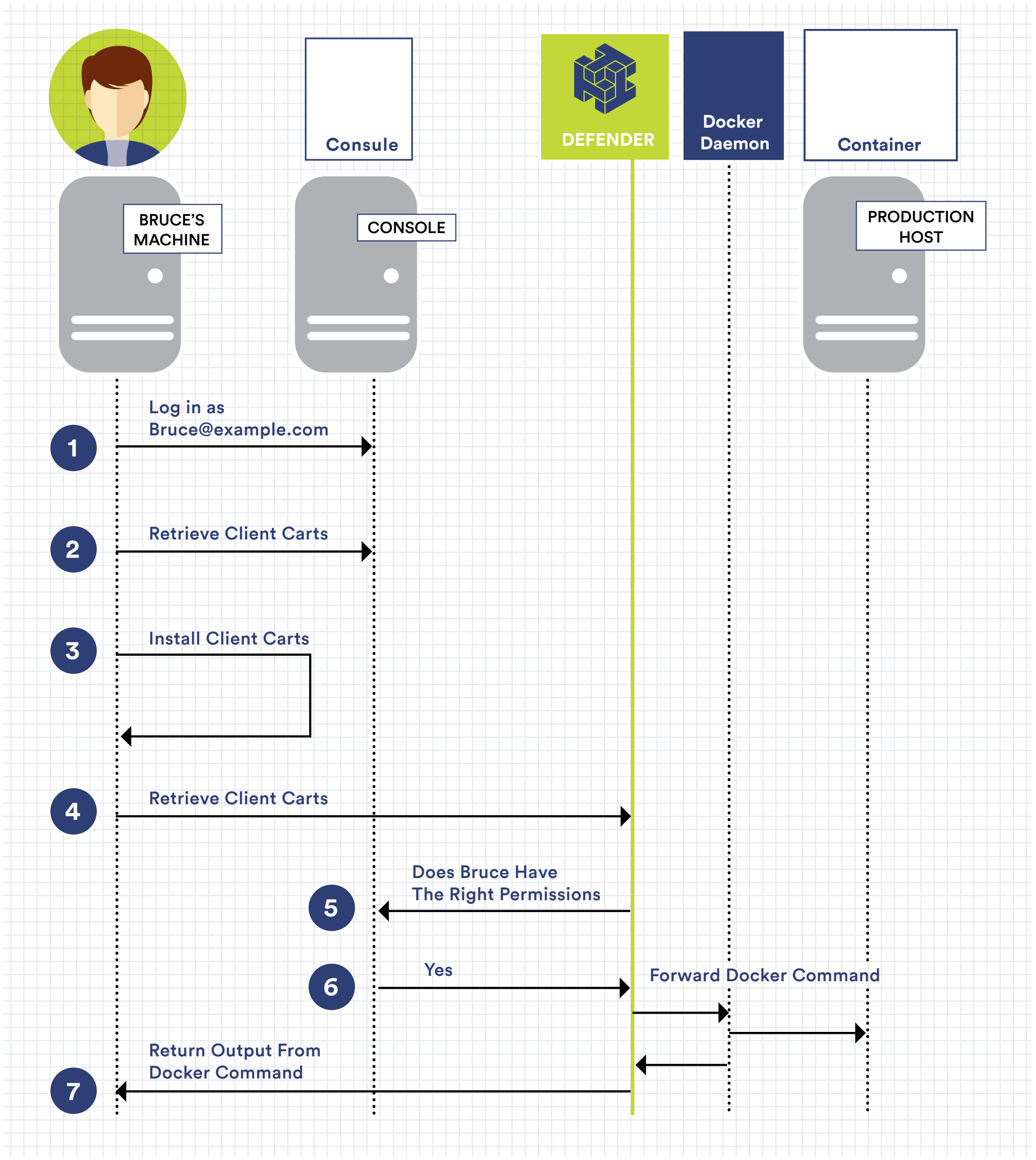
DISCUSSION

Users working with containers are authenticated with certificates. Certificates are bound to the user's identity. Users can download their certificates from Console.

NOTE: If you have already distributed certs to your users, Twistlock can be set up to use them instead.

To retrieve their certificates, users log into Console with their Active Directory, OpenLDAP, SAML, or Twistlock local user credentials. After installing their certificates on their host machine, users can access your container environment. When a user runs a Docker Engine, Docker Swarm, or Kubernetes command on a container protected by Defender, Defender authenticates the user, then applies the current access policy to either allow or block the command.

The following diagram shows how a developer, Bruce, downloads and installs his certificate and then runs a Docker command on a container running in the development environment.



TWISTLOCK CONFIGURATION GUIDE

No action required. As long as you have set up your users as described in Requirement 8.1.1, Twistlock manages the generation and distribution of certificates.

On the client side, anyone accessing the container environment must download and install their certificates from Console.

MORE INFORMATION

Configure Docker client variables
Using custom certificates for authentication

REQUIREMENT 8.5

Do not use group, shared, or generic IDs, passwords, or other authentication methods as follows:

- Generic user IDs are disabled or removed.
- Shared user IDs do not exist for system administration and other critical functions.
- Shared and generic user IDs are not used to administer any system components.

DISCUSSION
Twistlock ships with a default user that has administrative-level privileges (admin:admin). To prevent multiple users from sharing this account, delete it.

TWISTLOCK CONFIGURATION GUIDE
Delete the default admin user.

If you have integrated Twistlock with a directory service, such as Active Directory, OpenLDAP, or SAML:

1 | Specify a group from your directory service that has admin-level privileges.

2 | Log into Console with a user from the group that has admin-level privileges.

3 | Delete the admin account.

If you cannot integrate with a directory service, Twistlock lets you create local users and groups. To delete the default admin user with this setup:

1 | Log into Console with the default admin account.

2 | Create a user, and assign it the Administrator role.

3 | Log out of the admin account, and log in with the user you just created.

4 | Delete the default admin account.

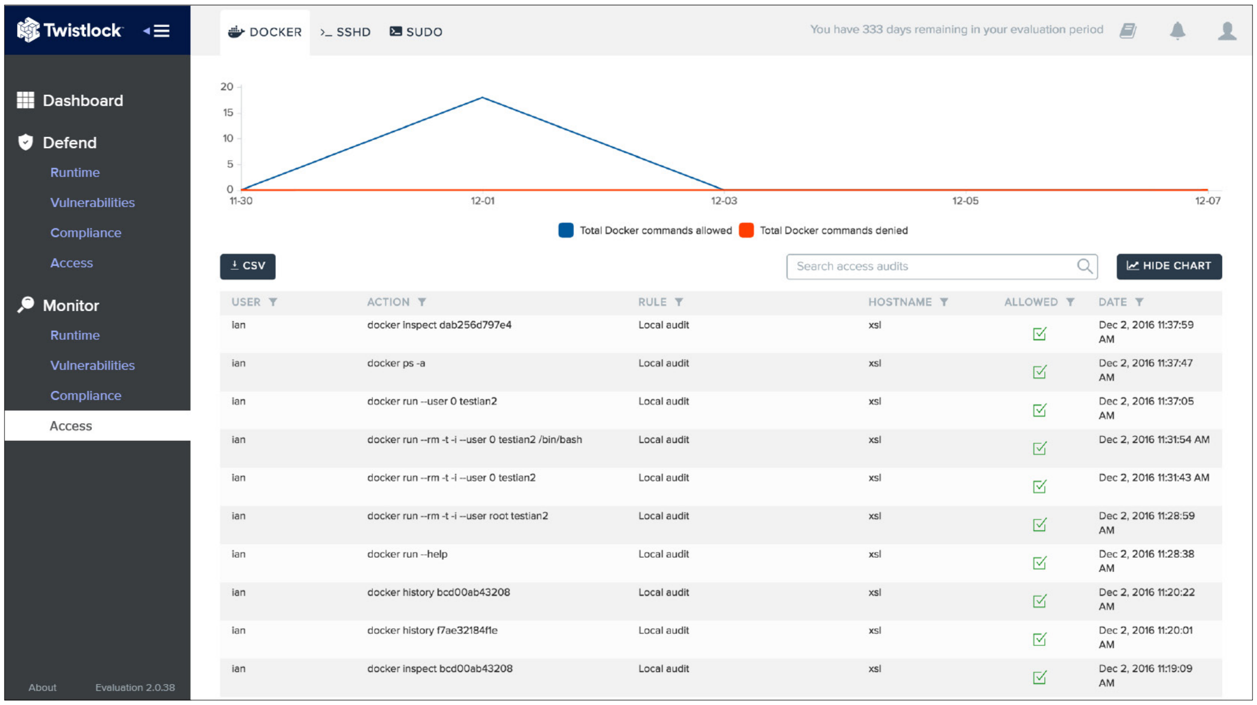
MORE INFORMATION
User roles
Assigning roles

REQUIREMENT 10.1

Implement audit trails to link all access to system components to each individual user.

DISCUSSION

Access to any container resource protected by Defender is logged and aggregated in Console. To view access events, log into Console, then go to Monitor > ACCESS. The following screenshot shows the access audit trail for Docker commands in Console.



USER	ACTION	RULE	HOSTNAME	ALLOWED	DATE
ian	docker inspect dab256d797e4	Local audit	xsl	✓	Dec 2, 2016 11:37:59 AM
ian	docker ps -a	Local audit	xsl	✓	Dec 2, 2016 11:37:47 AM
ian	docker run --user 0 testian2	Local audit	xsl	✓	Dec 2, 2016 11:37:05 AM
ian	docker run --rm -t -i --user 0 testian2 /bin/bash	Local audit	xsl	✓	Dec 2, 2016 11:31:54 AM
ian	docker run --rm -t -i --user 0 testian2	Local audit	xsl	✓	Dec 2, 2016 11:31:43 AM
ian	docker run --rm -t -i --user root testian2	Local audit	xsl	✓	Dec 2, 2016 11:28:59 AM
ian	docker run --help	Local audit	xsl	✓	Dec 2, 2016 11:28:38 AM
ian	docker history bcd00ab43208	Local audit	xsl	✓	Dec 2, 2016 11:20:22 AM
ian	docker history f7ae32184f1e	Local audit	xsl	✓	Dec 2, 2016 11:20:01 AM
ian	docker inspect bcd00ab43208	Local audit	xsl	✓	Dec 2, 2016 11:19:09 AM

TWISTLOCK CONFIGURATION GUIDE

No action is required. Access audits is enabled by default.

REQUIREMENT 10.3 (10.3.1, 10.3.2, 10.3.3, 10.3.4, 10.3.6)

Record at least the following audit trail entries for all system components for each event:

- 10.3.1: User identification
- 10.3.2: Type of event
- 10.3.3: Date and time
- 10.3.4: Success or failure indication
- 10.3.6: Identity or name of affected data, system component, or resource.

DISCUSSION
Twistlock logs all access events. The information recorded in the access event meets requirements 10.3.1, 10.3.2, 10.3.3, 10.3.4, and 10.3.6.

The following listing shows an example access event:

```
{
   “_id”: “5806a8e3423340f00168eff”,
   “ruleName”: “Local audit”,
   “allow”: true,
   “hostname”: “cto-dev-console-1.c.cto-sandbox.internal”,
   “time”: “2016-10-18T22:57:39.783Z”,
   “user”: “aqsa”,
   “action”: {
   “type”: “docker”,
   “verb”: “”,
   “resource”: “”,
   “namespace”: “”,
   “name”: “docker ps”
   }
}
```

WHERE:

- The user key contains the username (satisfies Requirement 10.3.1).
- The action object records the command the user ran (satisfies Requirement 10.3.2).
- The time field records the date and time of the user ran the command (satisfies Requirement 10.3.3).
- The allow field records whether the command was allowed or blocked according to your access policy. The specific rule is that allowed or blocked the command is recorded in ruleName (satisfies Requirement 10.3.4).
- The hostname key records the host where the command was run or blocked. (satisfies Requirement 10.3.6).

IMPLEMENTATION GUIDANCE
No action required.

REQUIREMENT 10.5.1

Limit viewing of audit trails to those with a job-related need.

DISCUSSION

In order to view audit events, you must log into Console. Only users with Admin, Operator, Defender Manager, and Auditor level privileges can view audit data in Console.

TWISTLOCK CONFIGURATION GUIDE

Assign the appropriate role to all groups that accesses Twistlock.

MORE INFORMATION

User roles

REQUIREMENT 10.5.2

Protect audit trail files from unauthorized modifications.

DISCUSSION
Audit events are stored in /var/lib/twistlock/db on the machine where Console is installed.

Because these files contains sensitive information, they are secured by setting ownership of the /var/lib/twistlock directory to root:root, and the permissions to 750 (writable by owner, readable by owner and group). Similarly, file ownership and group ownership for all data files is root, and permissions is set to 644 (writable by owner, readable by all).

TWISTLOCK CONFIGURATION GUIDE
No action required. File and directory permissions are automatically set up when Twistlock is installed.

REQUIREMENT 10.5.3

Promptly back up audit trail files to a centralized log server or media that is difficult to alter.

DISCUSSION

Twistlock, by itself, cannot satisfy this requirement. However, Twistlock can be configured to send all audit events to syslog in RFC5424-compliant format. If you already have a centralized syslog collector that backs up your log files, you can easily integrate Twistlock into your existing infrastructure to satisfy this requirement.

TWISTLOCK CONFIGURATION GUIDE

Configure Twistlock to send audit events to syslog.

MORE INFORMATION

Syslog integration

SAMPLE POLICY FILE

This guide comes with a sample policy file that can be used to deploy the recommended compliance checks by way of the Twistlock API.

The Twistlock API provides complete control over the product, and includes endpoints to install and deploy policies. The API gives you a way to formalize your policies in a file so that they can be reused, audited, and version controlled. Formalizing your policies this way ensures consistent configuration across your various environments.

The sample policy file, policy_pci.txt, installs a rule that blocks any container that violates compliance check 21, 41, 59, or 599. The compliance checks recommended in this guide are summarized in the following table:

PCI DSS	Twistlock compliance check ID	Action
Requirement 1.2	59 (CIS 5.9)	Block
Requirement 1.2.1	21 (CIS 2.1)	Block
Requirement 7.1.2	41 (CIS 4.1) 599	Block Block

Alerting and blocking
Twistlock ships with a default policy that raises an alert when any of the compliance checks fail. Alerting provides a benign way to validate and tune your policies on production systems. It lets you analyze your logs and reduce the number of false positives that could disrupt your service. After you are confident that your policies have the intended effect, you can flip the action from alert to block.

The block action in the sample policy file is for illustrative purposes only. It is used to differentiate the checks in the sample file (action=block) from the compliance rules in Twistlock's default rule (action=-block). Apply blocking only after you have validated your policies with alerts.

Deploying the policies in policies_pci.txt
This section shows you how to upload policy_pci.txt using the Twistlock API.
For each of the following steps, replace <CONSOLE> with the hostname or IP address for Console.

1 | Get your auth token.
The following command specifies the default admin username and password.
Specify the appropriate admin user for your environment.

a. **`$ curl -k -H “Content-Type: application/json” \`**

b. **`-d ‘{“username”:”admin”, “password”:”admin”}’ \`**

c. **`https://<CONSOLE>:8083/api/v1/authenticate`**

d. `{“token”:”eyJ0eXAiOiJKV1QiLC...”}`

2 | Update the installed policies by pushing the new JSON payload in policy_pci.txt.

a. **`$ curl -k -H “Authorization: Bearer eyJ0eXAiOiJKV1QiLC ...” \`**

b. **`-H “Content-Type:application/json” \`**

c. **`-X PUT \`**

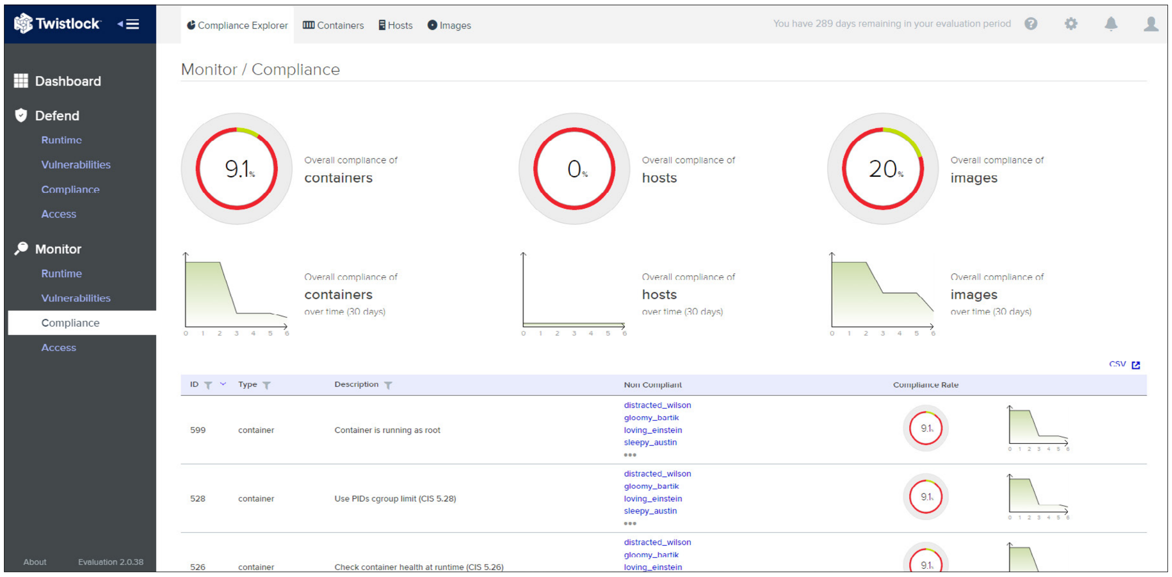
d. **`https://<CONSOLE>:8083/api/v1/policy/compliance \`**

e. **`--data-binary “@policy_pci.txt”`**

f. `policy updated`

3 | Log into Console, and go to **Defend > TRUST > COMPLIANCE.**

4 | Validate your new rule has been installed.


Twistlock
Compliance Explorer
Containers
Hosts
Images
You have 289 days remaining in your evaluation period
Dashboard
Defend
Runtime
Vulnerabilities
Compliance
Access
Monitor
Runtime
Vulnerabilities
Compliance
Access
About
Evaluation 2.0.38
Monitor / Compliance
9.1%
Overall compliance of containers
0%
Overall compliance of hosts
20%
Overall compliance of images
Overall compliance of containers over time (30 days)
Overall compliance of hosts over time (30 days)
Overall compliance of images over time (30 days)
CSV

ID	Type	Description	Non Compliant	Compliance Rate
599	container	Container is running as root	distracted_wilson gloomy_bartik loving_einstein sleepy_austin •••	9.1%
528	container	Use PIDs cgroup limit (CIS 5.28)	distracted_wilson gloomy_bartik loving_einstein sleepy_austin •••	9.1%
526	container	Check container health at runtime (CIS 5.26)	distracted_wilson gloomy_bartik loving_einstein	9.1%

Notes about the JSON in policy_pci.txt:

- Rules are processed in priority order. The first rule in the statement array has the highest priority. Therefore, any container that violates compliance check 59 is blocked, and the alert action specified in the default rule is overridden. The endpoint updates all compliance rules in a single shot to retain a strict order between rules.
- The name of the rule, as it appears in Console, is PCI compliance rules is specified with the name key.
- The value for last_modified is displayed in Console. It is a user defined value that can be used to track the version of the installed rule. It does not designate the time that the policy file was uploaded to Console.
- To create your own policy file from scratch, get the default policy from the /api/v1/policy/compliance endpoint, edit the JSON, then push the updated JSON back to the same endpoint. For more information, see the Twistlock API.

ABOUT TWISTLOCK

ENTERPRISE SECURITY. DEVOPS AGILITY.

Twistlock protects today's applications from tomorrow's threats with advanced intelligence and machine learning capabilities. Automated policy creation and enforcement along with native integration to leading CI/CD tools provide security that enables innovation by not slowing development. Robust compliance checks and extensibility allow full control over your environment from developer workstations through to production. As the first end-to-end container security solution, Twistlock is purpose-built to deliver modern security.

LEARN MORE AT

Twistlock.com