WHITE PAPER

RED HAT OPENSHIFT CONTAINER PLATFORM ARCHITECTURE DESIGN GUIDE FOR PCI DSS V3.2.1

ARCHITECTURE DESIGN GUIDE TO ASSIST CUSTOMERS IN PCI DSS V3.2.1 DEPLOYMENTS

JASON MACALLISTER
CHRIS KRUEGER | CISSP

FINAL V1.0

COALFIRE

North America | Europe
877.224.8077 | info@coalfire.com | Coalfire.com

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Red Hat, Inc. (Red Hat) delivers a comprehensive portfolio of products and services built from open source software components using an affordable, predictable subscription and support model. Red Hat has engaged Coalfire, a respected Payment Card Industry (PCI) Qualified Security Assessor Company (QSAC), to conduct an independent technical review of Red Hat OpenShift Container Platform (OpenShift or OCP) version 4.5 on Red Hat Enterprise Linux CoreOS (Red Hat CoreOS). The result of this review is provided in this Architecture Design Guide (ADG). This guide examines a use case for a payment entity's deployment of OpenShift in alignment with technical requirements to the Payment Card Industry Data Security Standard version 3.2.1 (PCI DSS version 3.2.1). The paper outlines Coalfire's methodology for assessment, the tangible approach used for the review; summarizes findings from our review of product capabilities, provides context into the possible use for this narrative, defines parameters to form a common basis of understanding, and opines as to the usefulness of OpenShift 4.5 within a program of compliance for PCI DSS version 3.2.1. This Coalfire ADG may be used to inform customer designs which intend to have self-assessment questionnaire (SAQ) or third-party Qualified Security Assessor (QSA) Report on Compliance (ROC) attestations performed.

This architecture design guide may be useful to any payment entity that is considering using OpenShift on Red Hat Enterprise Linux CoreOS as part of their cardholder data environment (CDE). This paper discusses segmentation strategies using OpenShift's software-defined networking (SDN) to isolate CDE systems from out-of-scope systems in an OpenShift environment. It further discusses control of data flow between containers or pods in an OpenShift environment using the OpenShift SDN's capabilities, e.g., between connected-to or security-impacting systems and the CDE. Finally, the paper outlines the PCI DSS version 3.2.1 requirements that are applicable to OpenShift on Red Hat Enterprise Linux and discusses and verifies the control implementation in the suggested design.

The review, findings, and opinions in this paper are only relevant to OpenShift 4.5 on RHEL CoreOS and are not applicable to any underlying hardware or platforms upon which OpenShift was deployed. General references to dependent infrastructure and adjacent technologies are mentioned to provide context for the deployed container platform.

COALFIRE OPINION

Security controls, features, and functionality that are built into OpenShift Container Platform on Red Hat Enterprise Linux CoreOS **can support and/or address** relevant technical PCI DSS version 3.2.1 requirements. OpenShift Container Platform provides granular control and improved security at scale for containerized workloads.

OVERVIEW OF PCI DSS VERSION 3.2.1

PCI DSS version 3.2.1 is a proprietary information security standard that was developed by the Payment Card Industry Security Standards Council (PCI SSC) to encourage and enhance cardholder data (CHD) security by stipulating a set of requirements regulating the use of information systems that handle CHD. PCI DSS is not an optional standard. As stated, all entities who process, store, or transmit CHD, regardless of geographic location, must comply with the standard or they can be fined and refused access to the card brand's payment system. To address OpenShift's usability by payment entities in PCI DSS relevant use cases, Coalfire utilized additional guidance provided by the PCI SSC including the Information Supplement: Guidance for PCI DSS Scoping and Network Segmentation May 2017, Information Supplement: PCI SSC Cloud Computing Guidelines April 2018, Information Supplement: PCI DSS Virtualization Guidelines June 2011, PCI DSS v3.2.1 Template for Report on Compliance, and other similar materials.

UNDERSTANDING PCI DSS SCOPE

A reliable approach for determining where PCI DSS is required to be applied begins with identification and definition of the scope of the entity's cardholder data environment (CDE) and PCI-DSS required, supporting systems for review. Per PCI DSS Requirements and Security Assessment Procedures, "PCI DSS security requirements apply to all system components included in or connected to the CDE. The CDE is comprised of people, processes, and technologies that store, process, or transmit cardholder data and sensitive authentication data." (PCI SSC, 2018). PCI DSS recommends that an assessed entity confirm the accuracy of their PCI DSS scope at least annually or prior to the annual assessment. To help identify scope, the payment entity should evaluate their systems to identify all locations and flows of CHD and identify all systems that are connected to or, if compromised, could impact the CDE. These systems should be included in scope for review.

For creation of Coalfire's ADG, the PCI DSS version 3.2.1 requirements were limited primarily to technical requirements pertaining to OpenShift on Red Hat CoreOS, including "network devices, servers, computing devices, and applications." (PCI SSC, 2018). The technical evaluation assumed the use of OCP 4.5 on Red Hat CoreOS for developing, testing, building, managing, monitoring, orchestrating, deploying, and hosting a payment entity's CDE applications. Categorizing the infrastructure and application components helps to identify the role that systems play with respect to the processing, transmission, and/or storage of CHD. PCI DSS provides three scoping categories: CDE systems, connected-to or security-impacting systems, and out-of-scope systems. Figure 1, taken from the PCI DSS Scoping and Segmentation Guide, illustrates how systems can be categorized and the factors for determining the appropriate category to assign.

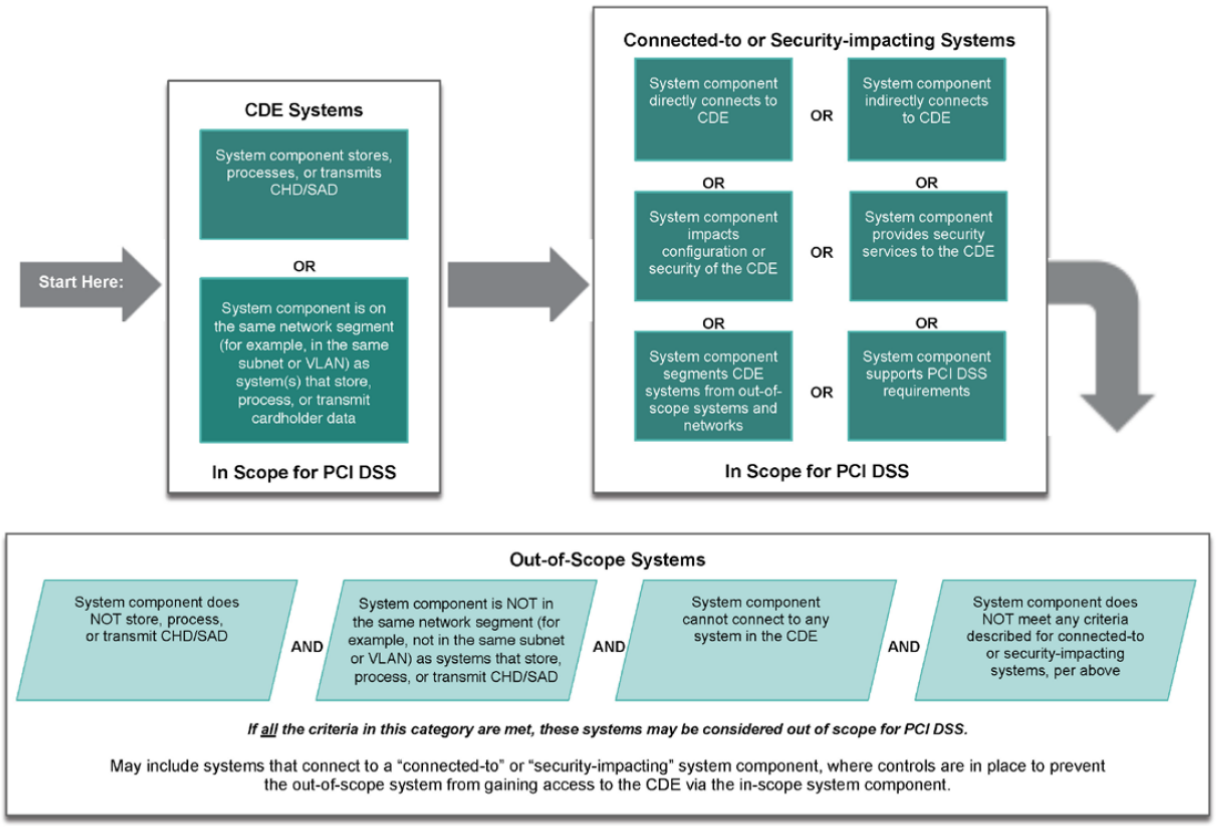

Start Here:
CDE Systems
System component stores, processes, or transmits CHD/SAD
OR
System component is on the same network segment (for example, in the same subnet or VLAN) as system(s) that store, process, or transmit cardholder data
In Scope for PCI DSS
Connected-to or Security-impacting Systems
System component directly connects to CDE
OR
System component indirectly connects to CDE
OR
OR
System component impacts configuration or security of the CDE
OR
System component provides security services to the CDE
OR
OR
System component segments CDE systems from out-of-scope systems and networks
OR
System component supports PCI DSS requirements
In Scope for PCI DSS
Out-of-Scope Systems
System component does NOT store, process, or transmit CHD/SAD
AND
System component is NOT in the same network segment (for example, not in the same subnet or VLAN) as systems that store, process, or transmit CHD/SAD
AND
System component cannot connect to any system in the CDE
AND
System component does NOT meet any criteria described for connected-to or security-impacting systems, per above
If all the criteria in this category are met, these systems may be considered out of scope for PCI DSS.
May include systems that connect to a "connected-to" or "security-impacting" system component, where controls are in place to prevent the out-of-scope system from gaining access to the CDE via the in-scope system component.

Figure 1 - PCI DSS Scoping Categories (PCI SSC, 2017)

Once the information or data is categorized, the payment entity can be better equipped to properly secure the data or apply necessary and required controls. Where data of differing categories is comingled, the level of security and control that must be applied is equal to the highest watermark for data in the information system. For instance, using PCI DSS data categories, if CDE systems, connected-to or security-impacting systems (C-T/SIS), and out-of-scope systems reside on the same flat network without network controls to limit connectivity between them, all systems, regardless of category, would require the same level of controls and safeguards to be applied. Network segmentation to isolate or control communication between categories of information and systems is useful for narrowing or minimizing the potential impact that one system could have on another.

NETWORK SEGMENTATION

Network segmentation is a recommended method for providing isolation for each category of system. According to PCI DSS version 3.2.1, isolating (segmenting) the CDE from the remainder of an entity's network is not a PCI DSS requirement; however, it is strongly recommended as a method that may reduce:

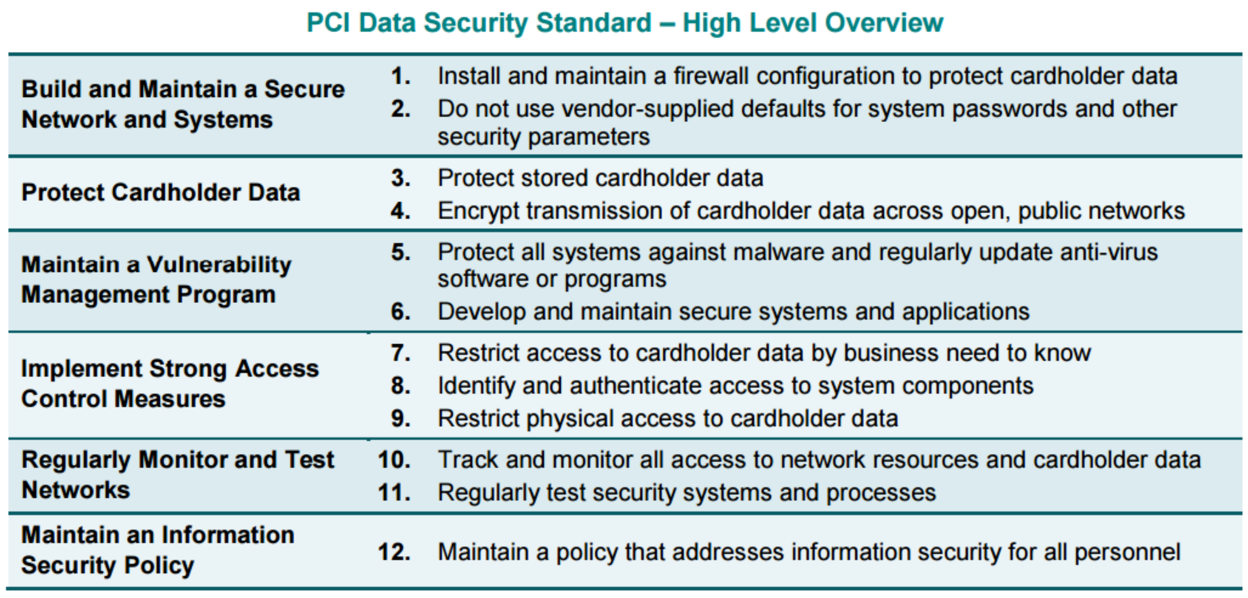
- The scope of the PCI DSS assessment
- The cost of the PCI DSS assessment
- The cost and difficulty of implementing and maintaining PCI DSS controls
- The risk to an organization (reduced by consolidating CHD into fewer, more controlled locations) (PCI SSC, 2018)

To evaluate the possibility for the use of segmentation to reduce scope, the payment entity must have a clear understanding of business needs and processes related to the storage, processing, or transmission of CHD. There is an assumption that risk to an organization is reduced through reduction and consolidation of CHD into fewer and more controlled locations as it pertains to storage, processing, and transmission. This is a fair assumption given that the reduction in scope decreases the surface area for attack and minimizes the number of entry points needed to be controlled.

The payment entity will need to consider the presence and placement of OpenShift and OpenShift components (control plane nodes, the Cluster Network Operator, the Ingress Operator, network policy objects, infrastructure pods, and so forth) within their overall infrastructure, especially as it relates to the use of network segmentation techniques to isolate CDE systems. Likewise, the payment entity will need to understand what categorization the components of OpenShift should have, as it relates to the role each component plays relative to the CDE systems. In that regard, each payment entity can benefit from understanding segmentation and control capabilities that are present within OpenShift on Red Hat CoreOS to enable the isolation of workloads. With OpenShift, workloads can be isolated to separate hosts in a container cluster as well as isolated on the network using OpenShift SDN capabilities. Current guidance from PCI SSC Cloud Special Interest Group strongly recommends separation of workloads representing differing zones of trust onto separate hosts. (Cloud Special Interest Group PCI SSC, 2018).

PCI DSS REQUIREMENTS

The PCI DSS standard is comprised of six “control objectives” with twelve “requirements”. The following is a listing of control objectives and their associated requirements. These technical and operational requirements were evaluated for applicability with the use of OpenShift on Red Hat CoreOS by a payment entity in a PCI DSS regulated environment. Ultimately, the burden for implementing PCI DSS requirements is on the payment entity, regardless of present capabilities of the technology to provide control.



PCI Data Security Standard – High Level Overview

Build and Maintain a Secure Network and Systems	**1.** Install and maintain a firewall configuration to protect cardholder data **2.** Do not use vendor-supplied defaults for system passwords and other security parameters
Protect Cardholder Data	**3.** Protect stored cardholder data **4.** Encrypt transmission of cardholder data across open, public networks
Maintain a Vulnerability Management Program	**5.** Protect all systems against malware and regularly update anti-virus software or programs **6.** Develop and maintain secure systems and applications
Implement Strong Access Control Measures	**7.** Restrict access to cardholder data by business need to know **8.** Identify and authenticate access to system components **9.** Restrict physical access to cardholder data
Regularly Monitor and Test Networks	**10.** Track and monitor all access to network resources and cardholder data **11.** Regularly test security systems and processes
Maintain an Information Security Policy	**12.** Maintain a policy that addresses information security for all personnel

Figure 2: PCI DSS High-Level Overview (PCI SSC, 2018)

NEW AND EMERGING TECHNOLOGY CHALLENGES TO COMPLIANCE

Containerization technologies can provide valuable benefits to the businesses that incorporate them into their service development and delivery process. Some of the benefits include increased developer productivity; decreased time to application deployment; increased application portability, agility, and scalability to align with changes in service demand; and increased compute efficiencies. Containerization technologies are beneficial to organizations looking to migrate to or expand their presence in the cloud. The compute efficiencies gained from containerization may potentially reduce the cost of cloud deployments over traditional bare-metal or virtualization deployments.

At the same time, challenges exist as it pertains to utilizing new and transformative technologies in the context of security compliance frameworks. Organizations seek to understand how introducing new technologies in their day-to-day operations may positively or negatively impact their security posture and/or compliance programs. It can be challenging to align security requirements to the new technology to address security application. Assessors who are responsible for measuring an entity's compliance program effectiveness may not be familiar with the technology. Traditional methods for security application may not be relevant or useful in the new environment.

Traditional anti-malware solutions, vulnerability scanning techniques, intrusion detection, and audit logging processes; for instance, may not be as effective in container run-time environments. The nature of how containers are executed on a container platform can limit the capabilities of traditional security tools to detect malware and identify vulnerabilities. However, with a proper understanding of container environments and the container lifecycle, new methods of applying security techniques to achieve the same or better outcome can be implemented. As an example, in most cases, containers that are deployed and executed in the container runtime are immutable. Containers are deployed from images. Images are stored in repositories or a registry waiting to be deployed and executed. With this knowledge, it seems reasonable to target scanning of vulnerabilities and malware at the image repository. In addition, as part of the systems development lifecycle, entities can establish a gate to inspect and digitally sign images prior to their placement in the image repository.

Many of the new and emerging technologies, such as containerization, software-defined data center, SDN, software-defined storage, and others, provide more efficient, scalable, extensible, and expedient means to support and deliver services for business and their customers. However, often the challenge with initial adoption of newer technologies is to understand and properly address the impact that a new technology has on security and compliance. A 2018 PCI SSC Special Interest Group's publication released in 2018 provided guidance on the use of many new technologies, with emphasis on cloud and new virtualization techniques, including containers; however, new technologies often emerge and evolve faster than regulating bodies are capable to address them. For organizations wishing to gain early benefits of a new technology, the level of risk must be evaluated and mitigated, especially as it may impact compliance requirements and the security of protected data.

SCOPE AND ADG REVIEW METHODOLOGY

Coalfire's understanding of OpenShift on Red Hat CoreOS and their combined capabilities was gained through product specification, installation, configuration, administration, and integration documentation provided by Red Hat and generally made publicly available from Red Hat's public-facing website. Coalfire further conducted interviews and engaged in live product demonstrations with Red Hat personnel. For live product demonstration purposes, OpenShift was deployed on Red Hat CoreOS in a lab environment to provide hands-on testing and analysis of the system's capabilities to support compliance.

Coalfire's review of OpenShift on Red Hat CoreOS began with a general alignment of the applicability of the technology against PCI DSS version 3.2.1 requirements. This was further narrowed down to specific requirements that were considered applicable to either OpenShift or the underlying operating system (OS). An analysis of capability for the reviewed technology to address the applicable requirements was then conducted. This analysis primarily focused on what an assessor might review when following the PCI DSS version 3.2.1 testing guidance during an assessment of applicable requirements.

Supporting material in the form of a previous PCI DSS 3.2-focused report may be found in the 2017 OpenShift Product Applicability Guide for PCI DSS version 3.2.. This document is being revised to incorporate both changes to OpenShift 4.5 and to the PCI DSS standard version 3.2.1.

Expanding on the general alignment, Coalfire engaged in a process of verification of control capabilities in a reference architecture designed and implemented by Red Hat. The verification used guidance from the PCI SSC when assessing PCI DSS as well as expectations for verification found in the companion Report on Compliance template.

SCOPE OF TECHNOLOGY AND SECURITY STANDARD TO REVIEW

Coalfire was tasked by Red Hat to review OpenShift as deployed on Red Hat CoreOS. The primary focus of the review included the components, features, and functionality of OpenShift along with the supporting underlying OS features and functionality when the components are deployed on Red Hat CoreOS. Workload pods and containers that were deployed in the lab environment were used for the purposes of demonstrating the platform's orchestration, deployment, and management capabilities. The pods and containers also served to support testing of segmentation within the environment to affirm that the deployed OpenShift SDN configuration was effective in isolating CDE from out-of-scope systems and to limit the connectivity of connected-to and security-impacting systems. Furthermore, Coalfire did not assess available image registries or repositories that may be used for acquiring applications, services, dependencies, or other elements to be hosted on or used within OpenShift.

For this review, Coalfire included requirements from the Payment Card Industry (PCI) Data Security Standard Requirements and Security Assessment Procedures Version 3.2.1, April 2016 publication available from https://www.pcisecuritystandards.org. For a broader understanding of the requirements and their applicability to technical solution implementation, Coalfire also reviewed supporting documentation provided by PCI SSC including controls assessment guidance, guidance on cloud and virtualization, scoping and segmentation guidelines, and other guidance that is made generally available from PCI SSC. Applied understanding of PCI DSS version 3.2.1 requirements and guidance was supplemented by documentation and guidance on relevant subjects. As OpenShift is a container platform, Coalfire also applied guidance from NIST Special Publication 800-190 Application Container Security Guide September 2017 publication as well as Information Supplement: PCI SSC Cloud Computing Guidelines April 2018 publication, specifically Appendix E.7 on Containers.

EVALUATION OF ARCHITECTURE

Coalfire evaluated the architecture to understand the components of an OpenShift implementation and the role these components have in the overall security of the OpenShift platform as well as the security capabilities that are extended to the running containers or workloads on the platform. Coalfire also evaluated the architecture to understand the capabilities of OpenShift on Red Hat CoreOS to properly isolate identified workloads according to their security categorization.

OPENSHIFT CONTAINER PLATFORM AND ARCHITECTURE

OpenShift offers a consistent hybrid cloud foundation for building, deploying and scaling containerized applications. OCP delivers a single, consistent Kubernetes platform anywhere Red Hat CoreOS runs. It is an integrated platform to run, orchestrate, monitor, and scale containers. OCP provides both streamlined automated installations as well as options for more customized installations where organizations have requirements not met by the automated defaults. OCP allows organizations to control, defend, and extend the application platform throughout an application's lifecycle. It also enables a secure software supply chain to make applications more secure.

OCP helps maximize developer productivity with specially configured toolsets and tools. One of these tools is a workflow that includes built-in Continuous Integration/Continuous Delivery (CI/CD) pipelines and a source-to-image capability that enables developers to go directly from application code to container. These updated toolsets help provide consistent operations and management experience across infrastructures and in support of many teams.

The following architectural elements are specific to each deployed cluster of OpenShift Container Platform. Coalfire has highlighted the elements that were most instrumental in supporting security and compliance for PCI DSS version 3.2.1. The deployed architecture is intended to serve as an example of how to deploy and utilize OpenShift in a manner that can be supported in the context of PCI DSS compliance.

For relevance to PCI DSS version 3.2.1, Coalfire and Red Hat considered the use case of a payment entity deploying CDE system components in an OpenShift environment in support of an e-commerce system. The OpenShift environment was inclusive of workloads representative of CDE systems, out-of-scope corporate LAN systems, and the C-T/SIS systems in-scope and servicing both the CDE and corporate LAN.

OPENSHIFT CONTAINER PLATFORM COMPONENTS AND ROLES

The following is a listing of components and roles that support OpenShift and were part of the deployed design:

- **Red Hat CoreOS** – OCP uses Red Hat CoreOS, a container-optimized operating system specifically configured for running the container platform. Red Hat CoreOS is installed and managed as part of OpenShift similarly to an appliance.

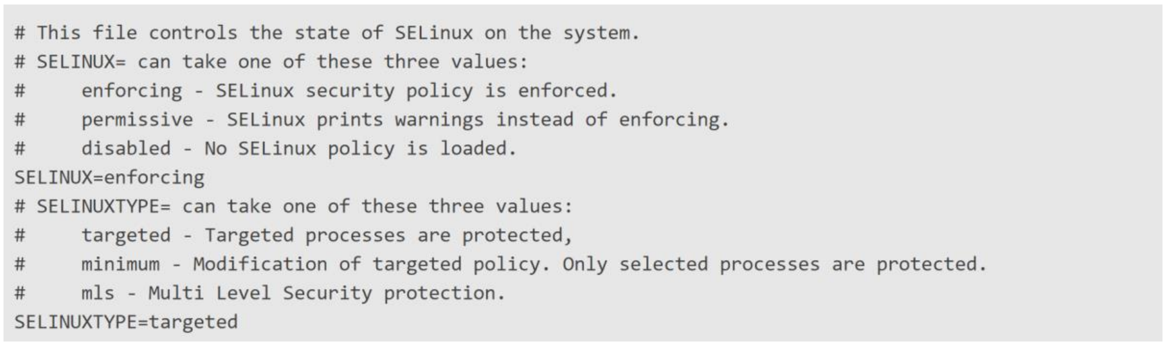
  Red Hat CoreOS is a single-purpose, container- and Kubernetes-optimized, minimal-footprint OS powered by the same binaries as RHEL. This OS can assist organizations with meeting requirements for system hardening with the least functionality through its lightweight, purpose-built nature; it only includes the necessary features, functions, and services to host containers in an OCP environment.

  Red Hat CoreOS is designed to be more tightly managed than a default RHEL installation. Management is performed remotely from the OCP cluster. When the OCP cluster is set up and Red Hat CoreOS is deployed, customers can only modify a few system settings.

Red Hat CoreOS has built-in security features and functionality that, as configured in an OCP installation, provide a secure platform for supporting the OCP components and the workloads in containers that OCP orchestrates.

For this PCI DSS studied implementation, as outlined in this design guide, OpenShift was deployed on Red Hat CoreOS. The following features of Red Hat CoreOS are configured by default. These features align with container orchestration recommendations from PCI SSC in PCI SSC Cloud Guidelines version 3.

- **Linux namespaces** are a fundamental aspect of containers in Linux and are used for creating an abstraction of a particular global system resource to make it appear as a separate instance to processes within the namespace. The kernel provides isolation by creating separate namespaces for containers. The types of namespaces used by containers in Red Hat CoreOS include mount namespaces, UTS namespaces, IPC namespaces, PID namespaces, and network namespaces.
- **SELinux** (Security-Enhanced Linux as delivered in RH CoreOS) provides an additional layer of security to keep containers isolated from each other and from the host. SELinux enforces mandatory access control (MAC) for every user, application, process, and file on a Linux system. It provides secure separation of containers by applying SELinux policy and labels. **SELINUX=enforcing** and **SELINUXTYPE=targeted** is a default in Red Hat CoreOS and configured in the **/etc/selinux/config** file (which is now a soft link to **/etc/sysconfig/selinux** in Red Hat CoreOS) as shown here:



```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Figure 3 - /etc/selinux/config

- **CGroups** (control groups) are used to limit, account for, and isolate the resource usage (CPU, memory, disk I/O, network, etc.) of a collection of processes. These are used to ensure that containers on the same host will not be impacted by other containers. CGroups allocate CPU time, system memory, network bandwidth, or combinations of these among user-defined groups of tasks.
- **Secure computing mode** (seccomp) profiles are used with containers to restrict available system calls. By defaut, OCP runs containers unconfined by seccomp. OCP can be configured to apply seccomp profiles.

Red Hat CoreOS is a container optimized OS, designed to support OCP. Red Hat CoreOS is deployed and managed as part of OpenShift. When the OCP cluster is setup and Red Hat CoreOS is deployed, customers can only modify a few system settings.

- **Operating Environment** – OpenShift can be deployed on bare-metal physical hardware, VMware vSphere, Red Hat Virtualization (RHV), OpenStack, or other major cloud providers. It can be deployed on private or certified public cloud environments, depending on the organization's specific use cases.
- **Open Container Initiative (OCI) Runtime** – Container Runtime Interface-Orchestration (CRI-O) is an OCI-compatible runtime that is installed on every Red Hat CoreOS host. As such, CRI-O enables the use of OCI-compatible containers. OCI has an open governance structure used to create open industry standards around container formats and runtimes. The CRI-O engine focuses on features needed by Kubernetes platforms, such as OCP, and offers specific compatibility with different Kubernetes versions.
- **Kubernetes** – Kubernetes is an open-source container orchestration engine for automating the deployment, scaling, scheduling, and management of containerized applications across the cluster. Kubernetes provides orchestration for complex multi-container services – serving as the de facto standard for orchestrating containers.
- **Containers** – End-user application instances, application components, or other services are run in Linux containers. A container should only include the necessary libraries, functions, elements, and code required to run the application.
- **Pods** – While application components run in containers, OCP orchestrates and manages pods. A Kubernetes pod is a group of containers that are deployed together on the same host. A pod is an orchestrated unit in OCP made up of one or more containers. A pod should only contain a single function, such as an application server or web server, and should not include multiple functions such as both a database and application server.
- **Operators** – An Operator is a method of packaging, deploying, and managing a Kubernetes-native application. Operators automate the lifecycle management of containerized applications within Kubernetes. A Kubernetes-native application is an application that is both deployed on Kubernetes and managed using the Kubernetes APIs and kubectl tooling. A controller is a core concept of Kubernetes. It is implemented as a software loop that runs continuously, compares, and, if necessary, reconciles the expressed desired state and the current state of an object. Objects are resources like Pods, Services, ConfigMaps, or PersistentVolumes; but, through Custom Resource Definitions (CRDs) could be any object. Operators apply the model of controller at the level of entire applications and are, in effect, application-specific custom controllers.

  OCP adds developer- and operations-centric tools to Kubernetes that help enable rapid application development, simplified deployment, and scaling, and long-term lifecycle maintenance for applications. OCP also leverages integrated components to automate application builds, deployments, scaling, and health management.

Components Used to Construct the Suggested Design

The following components are essential elements of OCP 4.5:

- **OpenShift Operators** – As OCP is a fully containerized platform that consists of many different components, OCP takes advantage of Operators for driving the installation, configuration, management, and upgrades of OCP and all its services. Operators are both the fundamental unit of the OCP 4.5 cluster and a way to deploy applications and software components for the OCP customer's applications to use. Operators deploy and manage OpenShift's native functions including the Kubernetes core services along with monitoring (e.g., Prometheus), logging (e.g., Elasticsearch, Fluentd, Kibana), software-defined networking, storage, registry. Operators serve as the platform foundation that removes the need for manual upgrades of the OSs and OCP control plane applications. The Cluster Version Operator and Machine Config Operator allow simplified cluster-wide management of those critical components. All the components of the platform are managed throughout their lifecycle with Operators.
- **Operator Lifecycle Manager** – The Operator Lifecycle Manager (OLM) is the backplane that facilitates the management of Operators on a Kubernetes cluster. OLM helps administrators of the cluster control which Operators are available in which namespaces, and who can interact with the running Operators. The permissions of an Operator are configured automatically to follow a least-privilege approach. The OLM helps users install, update, and manage the lifecycle of all Operators and their associated services running across their clusters. The OLM runs by default in OCP 4, which aids administrators in installing, upgrading, and granting access to Operators running on their cluster. Because OpenShift Container Platform is fundamentally composed of components deployed and managed by operators, OLM also facilitates the upgrading of the entire OpenShift Container Platform cluster.
- **Machine Config Operator (MCO)** – The MCO manages OS updates and OS configuration changes. The MCO allows platform administrators to ensure consistent configuration across all Red Hat CoreOS nodes, monitoring for drift, and alerting on unsupported configurations.
- **OpenShift Worker Nodes** – In the default deployment, 2 worker nodes represent instances of Red Hat CoreOS with the OCP 4.5 software deployed. Worker nodes are where end-user applications and workload infrastructure components, such as routers, run in containers. Worker nodes will contain the necessary OCP node daemon, the kubelet, the container runtime, and other services required to support the hosting of containers. Most of the software components that run above the OS (e.g., the software-defined network [SDN] daemon) run in containers on the Nodes.

  Worker node configurations are bootstrapped at install time. The worker nodes advertise their capacity and the scheduler, which is part of the master services, determines on which nodes to start containers and pods. Important services run on each worker node, including CRI-O (the container engine), kubelet (the service that accepts and fulfills requests for running and stopping container workloads), and a service proxy (manages communication for pods across workers). The quantity of worker nodes is configurable at deployment and may be easily scaled up to increase application capacity and performance.
- **OpenShift Control Plane Nodes** – In the default deployment, 3 nodes comprise the control plane for OCP. The control plane maintains and understands the state of the cluster and orchestrates all activity that occurs on the worker nodes. The OCP Control Plane nodes are run on Red Hat CoreOS. While it is possible to configure the Control Plane nodes to run application instances, for separation of function, application instances (pods) should not be scheduled on Control Plane nodes. The following are the four functions of the OCP control plane:

  - **API and Authentication** – The Control Plane provides a single API where all tooling and systems interact. Everything that interacts with OCP must go through this API. All API requests are Transport Layer Security (TLS) encrypted and must be authenticated. Authorizations are handled by fine-grained role-based access control (RBAC). It is recommended to tie the Control Plane to an external identity and access management system using Lightweight Directory Access Protocol (LDAP), OpenID Connect, or other providers. The Control Plane evaluates requests for both authentication (AuthN) and authorization (AuthZ). Users of OCP who have been granted access can be authorized to work with specific projects.
  - **Desired and Current State** – The state of OCP is held in the OCP data store. The data store uses etcd, a distributed key-value store. The data store houses information about the OCP environment, including OCP user account information and the RBAC rules, the OpenShift environment state, application environment information and important environment variables, secrets data, and other information.
  - **Scheduler** – The scheduler determines pod placement within OCP. It uses a combination of configuration and environment state (CPU, memory, and other environmental factors) to determine the best fit for running pods across the Nodes in the environment. The scheduler is configured with a JSON file in combination with Node labels to carve up OCP resources. This allows the placement of pods within OCP to be based on the real-world topology, making use of concepts such as regions, zones, or other constructs relevant to the enterprise. These factors can contribute to the scheduled placement of pods in the environment and can ensure that pods run on appropriate Nodes associated with their function.
  - **Health and Scaling** – The OCP Control Plane is also responsible for monitoring the health of pods and scaling the pods as desired to handle additional load. The OCP Control Plane executes liveness and readiness tests using probes that are defined by users. The OCP Control Plane can detect failed pods and remediate failures as they occur.

- **Service Layer** – The OpenShift Service Layer allows for application components to easily communicate with one another. For instance, a front-end web service containing multiple web servers would connect to database instances by communication via the database service. OpenShift automatically and transparently handles load balancing across the services' instances. In conjunction with health probes, the OpenShift Service Layer ensures that traffic is only directed toward healthy pods, which helps to maintain component availability.
- **Persistent Storage** – Linux containers are natively ephemeral and only maintain data for as long as they are running. Applications and/or application components may require access to a long-term persistent storage repository, such as may be required for a database engine. OpenShift provides the means to connect pods to external real-world storage, which allows for stateful applications to be used on the platform. Persistent storage types that are usable include iSCSI, Fiber Channel, and NFS, as well as cloud-type storage and software-defined storage options such as Red Hat OpenShift Container Storage. Persistent storage can be dynamically provisioned upon the user's request, provided the storage solution has an integration with OpenShift.
- **Ingress Controller** – The Ingress Controller is commonly used to allow external access to an OCP cluster. The Ingress Operator manages Ingress Controllers. An Ingress Controller is configured to accept external requests and proxy them based on the configured routes. External requests are limited to HTTP and HTTPS using Server Name Indication (SNI) and Transport Layer Security (TLS), which is enough for web applications and services. Ingress works in partnership with the service layer to provide automated load balancing to pods for external clients. The Ingress Controller uses the

service endpoint information to determine where to route and load balance traffic; however, it does not route traffic through the Service Layer.

- **OpenShift SDN** – The OCP software-defined network (SDN) is a unified cluster network that enables the communication between pods across the OCP cluster. The OCP SDN configures an overlay network that uses Open vSwitch (OVS). Red Hat currently provides one SDN mode for the OCP pod network – the networkpolicy mode. The network policy mode provides fine-grained access control via user-defined rules. Network policy rules can be built in a mandatory access control (MAC) style, where all traffic is denied by default unless a rule explicitly exists, even for pods/containers on the same host.

- **OpenShift Registry** – The OpenShift Registry provides integrated storage and management for sharing container images. OpenShift can utilize existing OCI-compliant container registries accessible to the Nodes and the OpenShift Control Plane via the network.

  Figure 4 below provides a high-level OpenShift Container Platform Overview.

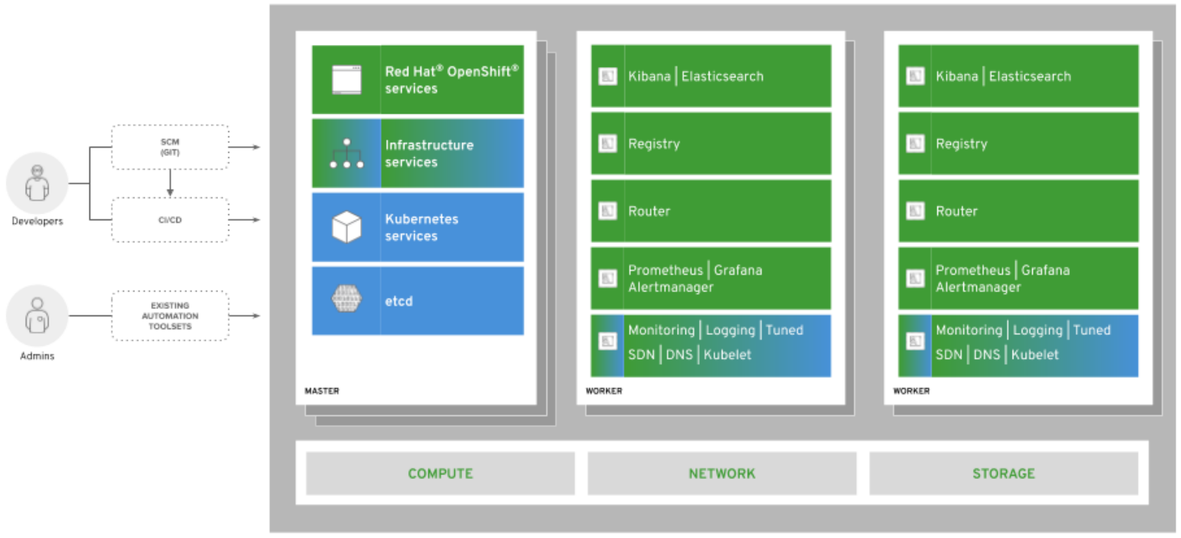


Figure 4: High-level OpenShift Container Platform Overview

Projects – A project is a Kubernetes namespace with additional OCP annotations and metadata. It is the central vehicle by which access to regular users' resources is managed and is the tenancy model of OCP and Kubernetes. A project allows a community of users to organize and manage their content in isolation from other communities.

Types of Users

When it comes to direct use and management of an OpenShift cluster, regular users (who typically run workloads and may do some administration) and system users (who can interact with the API), and service accounts used for automation purposes.

- **Users** – User (operators, developers, application administrators) access to OCP is provided through standard interfaces, including the Web UI, CLI, and IDEs. These interfaces go through the authenticated and RBAC-controlled API. Users do not require system-level access to any of the OCP hosts, even for complex application debugging and troubleshooting tasks.

Three types of users can exist in an OCP environment: regular users, system users, and service accounts.

- – **Regular Users:** A user most commonly represents a real person who interacts with the OCP cluster in some way. Since the OCP management is performed on the API, this includes both administrators of the cluster and regular cluster users who run their workloads.
- – **System Users:** Many of the system users are created automatically when the OCP infrastructure is defined to enable the infrastructure to interact with the API securely. System users include a cluster administrator (with access to everything), a per-node user, users for use by ingress controllers and registries, and various others. There is also an anonymous system user that is used by default for unauthenticated requests. All unauthenticated requests are subject to authorization. Examples of these users are system:admin, system:openshift-registry, and system:node:node1.example.com.
- – **Service Accounts:** These are non-human system users, often associated with projects and used for API access in automation situations. Some default service accounts are created and associated when a project is first created. Project and cluster administrators can create additional service accounts for defining access to the contents of each project.

For more information on OpenShift concepts, features, and functions, please refer to Red Hat's product documentation.

Architecture Design

This PCI DSS design pattern was implemented following Red Hat's recommendation for production OpenShift environments where two MachineSets were created during installation. The master MachineSet pertains to the control plane assets, while the worker machine set is created to support user applications. Worker role machines drive compute workloads that are governed by a specific machine pool that autoscales them. These worker machines are classed as *compute* machines. Custom machine sets can be created by cluster-admins. Node selectors or references to MachineSets can be used to determine the placement of pods onto nodes using the scheduler.

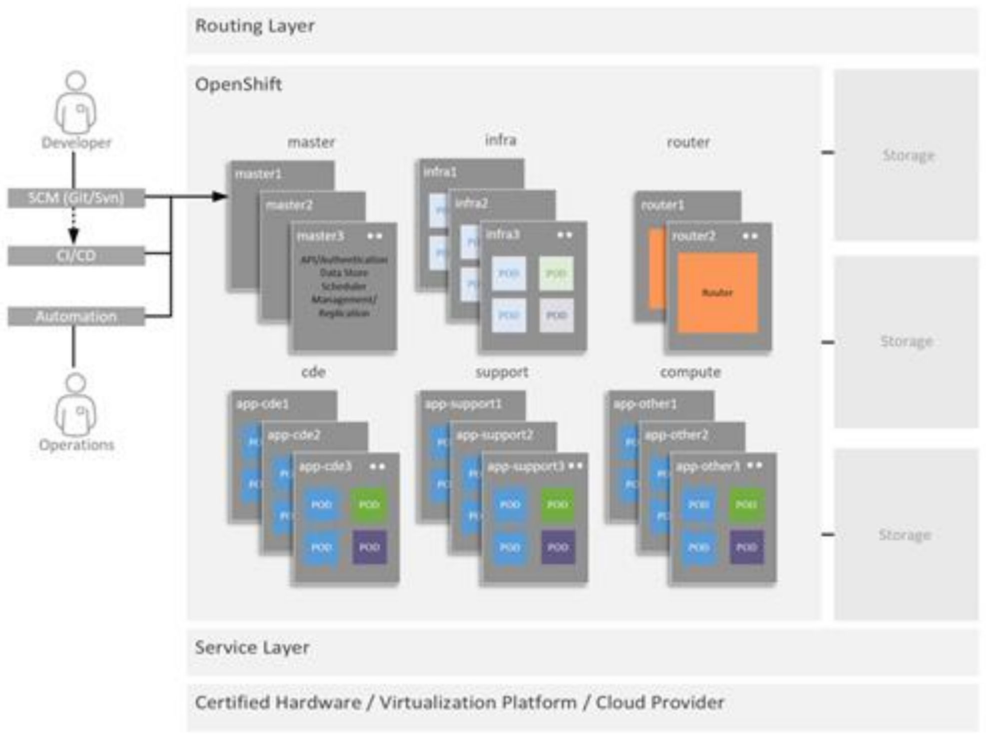
Using a default configuration, the cluster was deployed with three Control Plane nodes to provide high availability (HA) for both the control plane and the etcd datastore. The studied environment used an additional load balancer (HAProxy) in front of the control plane to separate access to the control plane. It should be noted that this control plane load balancer is completely different from the OpenShift router even though they are both implemented using HAProxy.

- **OpenShift Machine Sets**

For this PCI DSS suggested implementation, additional distinct MachineSets are created to represent PCI DSS v3.2.1 scoping security categories. The cardholder data environment (CDE) nodes are assigned a role with the label **cde**, the connected-to and security-impacting systems are assigned a role with the label **support**, and the out-of-scope systems are assigned the default role with the label **compute**. Node labels were assigned to each of the node groups.

Nodes for the Control Plane have their own specific node groups. Additional MachineSets were created for further segmenting the OpenShift infrastructure components - specifically, the OpenShift router to the pods and log aggregation and cluster metrics collection solutions included with the platform. The nature of the OpenShift components that run on these nodes relative to the CDE systems also make them in scope for PCI DSS assessment as the category connected-to and security-impacting systems.

Figure 5 illustrates these node groupings, the naming of the nodes in each of the groups, and notionally the deployed pods that are running on each of the nodes. The alignment of application pods and containers by security category to the separate node groupings served to provide host-level isolation for the workloads. The diagram also illustrates the relationship of OpenShift to the underlying and surrounding or supporting infrastructure. The depiction of the diagram of Developer and Operations illustrate the administrative connectivity to the environment for management and operational support of OpenShift and the workloads.



The PCI DSS design pattern includes three master nodes, three dedicated infrastructure nodes, three compute nodes, three **cde** nodes, three **support** nodes, and two router nodes. Dedicated infrastructure nodes and router nodes are used to allow registry, logging, metrics and router pods to run on hosts separately from pods used for user applications.

Figure 5: CoreOS OCP Architecture

Proper node labeling and project node selector definitions ensure workloads land on appropriate hosts. Two methods of controlling workload residence on particular nodes may be used in OCP 4.5: A legacy method formerly the choice for OCP 3 designs – default node selection via the editing of the Scheduler Operator **defaultNodeSelector** variable in the **spec:** section; or the preferred newer method employing Taint / Toleration approaches to regulate pods onto desired hosts.

A Taint is a kind of labeling that allows a host to repel a set of pods that do not have the capability to run on them. Applied through the node specification (**NodeSpec**), Taints may be used to ensure that workload pods avoid control-plane hosts and to select appropriate worker nodes in desired regions.

A Toleration is a way to overcome a taint. By editing the pod specification (**PodSpec**) the developer may create exceptions to the Taint model, should those be desired.

With this new approach, OCP implementation recommendations are to use the Taint labeling is to create host allocation to support the PCI DSS segmentation objectives for pods by specifically tainting hosts which are intended to run CDE and connected-to/security-affecting workloads with a unique label to enforce segmentation requirements for later PCI DSS assessment. In a similar way, restriction of out-of-scope pods can be enforced with taints on hosts used for non-CDE purposes.

Toleration labeling on pods could also be used to illustrate enforcement onto appropriate hosts, or to create exceptions, perhaps for staging or test pods to temporarily run on out-of-scope tainted hosts, for instance.

The Taint/Toleration constructs are powerful ways to enable proper project node selection and are the currently recommended approaches to ensure enforcement of pod allocation to desired nodes and to create an implementation that may be easily reviewed by PCI DSS QSAs.

Through RBAC assignments, activities can be restricted for OpenShift administration, project administration, and workload administration. For instance, the cluster is configured so that only cluster admins are capable of creating projects with Taints/Tolerations or NodeSelectors that will land workloads on the **cde** or **support** nodes.

When setting up the OpenShift environment for PCI DSS deployments, a set of users should be configured to be production admins with restrictions to allow these users to deploy/modify workloads in the CDE/support environments.

User Workloads

Much like distinct node groups are defined according to PCI DSS security category, user workload containers and pods are deployed in the studied design to represent the different security categories. The deployed workload notionally represents an e-commerce system with payment components. Additional workloads were deployed to represent out-of-scope systems and connected-to and security-impacting systems. While the workloads in an OpenShift environment may be handled by the payment entity according to applicable and relevant security requirements, it is generally understood that the security features and functions of the OpenShift environment are universally applied regardless of the running workload.

Special care should be taken when deploying privileged containers. For the most part, the use of privileged containers should be limited to special circumstances where such privileges are required for successful operation of the container. If possible, custom security context constraints (SCC) with only the required privileges should be created to minimize the possibility for security issues. Privileged containers should additionally be restricted to specific nodes to avoid possible security risks, should such containers be compromised. In general, the use of the explicit privileged SCC should be avoided at all costs for security reasons.

Managing Ingress

The OpenShift router was placed on dedicated router nodes as shown in Figure 6. The routers support ingress to the applications hosted in OpenShift and sit between the public/DMZ network and the internal container network. Router pods are only deployed to node hosts with the Taint label of **infra,** and with

Toleration PodSpecs for **CDE**. Figure 5 depicts utilization of dedicated router nodes with router pods to provide isolation between the network external to OpenShift and the internal OpenShift SDN. Ingress traffic flows in through OpenShift's routing layer to ingress routers and then is directed appropriately to the targeted workload. This figure also depicts the optional use of unique router nodes and pods for handling of Internet traffic separate from intranet traffic where Internet router nodes and pods are connected to the DMZ.

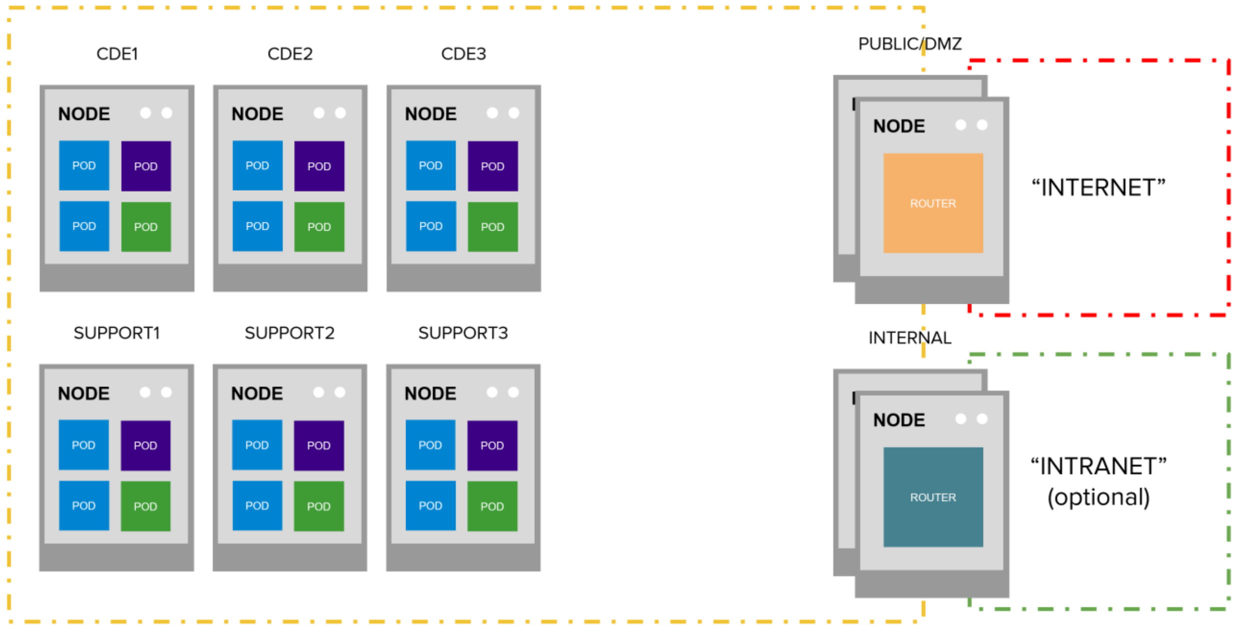


Figure 6 - OpenShift Router Configuration

Ingress for workloads (pods/containers) in the environment flow through dedicated router pods deployed in this design on dedicated router nodes. The private network depicted in Figure 6 represents the virtual or physical network attached to the OpenShift hosts. This network supports node communication to external services such as DNS, LDAP, and NTP as well as provides connectivity for users to the API, CLI, and web console.

OpenShift DNS

OpenShift Container Platform includes an internal DNS subsystem that provides hostname resolution as it relates to Kubernetes Services. Using service hostnames allows for flexibility in deployment across different software development lifecycle (SDLC) environments, which helps to prevent misconfigurations when promoting workloads.

OpenShift SDN and Network Policies

The SDN provides extremely fine-grained access control via user-defined network policy rules. Network policy rules can be built in a "mandatory access control" style, where all traffic is denied by default unless a rule explicitly exists, even for pods/containers on the same host.

The studied implementation utilizes network policy rules for fine grained control for supporting approved traffic between pods associated with different projects and between pods within the same project. Network policy rules also allowed for default policy enforcement to block all traffic regardless of source with traffic only being allowed to the destination by explicit policy declaration.

A recap of PCI DSS requirements illustrate that the basic segmentation best practices are to be created in the architecture, as shown in figure 7. These rules are summarized here: out-of-scope corporate LAN cannot communicate with **Support/Infrastructure** or **CDE** nodes and pods; connected-to/security affecting pods (**Support/Infrastructure**) can communicate with each other and with the **CDE** nodes and pods; and obviously nodes and pods of the same type can communicate with each other.

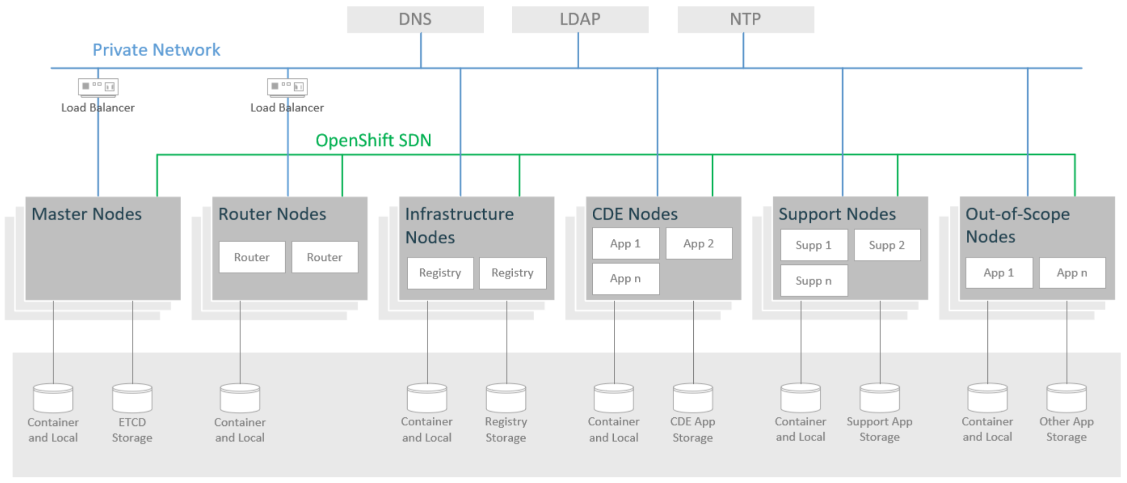


Figure 7 - High-level OCP Network Topology

On an OpenShift Control Plane, OpenShift SDN maintains a registry of nodes, stored in etcd. When a node is registered into the cluster, OpenShift allocates an unused subnet from the cluster SDN and stores this subnet in the registry. When a node is deleted, OpenShift deletes the subnet from the registry and considers the subnet available to be allocated again. Control PlaneControl PlaneControl Plane

On the node, once the OpenShift SDN registers the local host with the SDN Control Plane, the OpenShift SDN creates three network devices: **br0**, **tun0**, and **vxlan_sys_4789.**

br0 is the OVS bridge device that pod containers will be attached to. OpenShift SDN also configures a set of non-subnet-specific flow rules on this bridge.

tun0 is an OVS internal port (port 2 on **br0**). This gets assigned the cluster subnet gateway address and is used for external network access. OpenShift SDN configures **netfilter** and routing rules to enable access from the cluster subnet to the external network via NAT.

vxlan_sys_4789 is the OVS VXLAN device (port1 on **br0**), which provides access to containers on remote nodes. This is also referred to as **vxlan0** in the OVS rules.

Each time a pod is started on the host, OpenShift SDN assigns the pod a free IP address from the node's cluster subnet, attaches the host side of the pod's veth interface pair (veth0/1) to the OVS bridge **br0**, and adds OpenFlow rules to the OVS database to route the traffic addressed to the new pod to the correct OVS port. The ovs-networkpolicy plug-in allows for NetworkPolicy rules to be injected into **br0** where the rules can be applied to packets destined for the target pod.

Figure illustrates the flow of traffic from a pod on one node to a pod on a remote node in the cluster. Traffic policy enforcement is applied at the OVS bridge device. In this case traffic will traverse the network outside of the node over vxlan0.

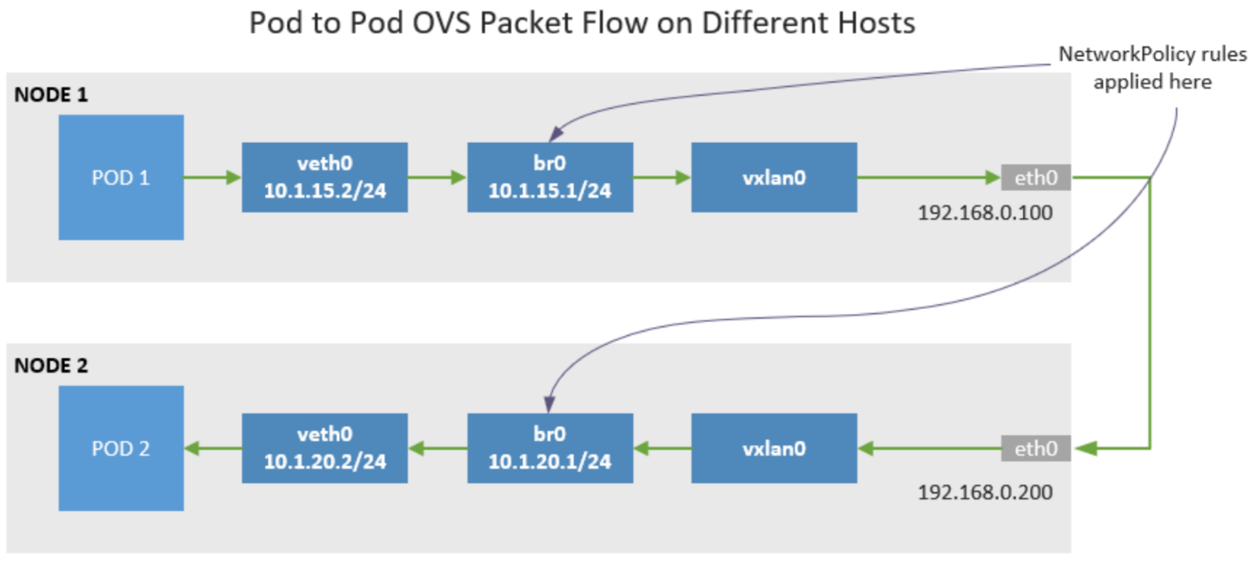


Figure 8 - OpenShift SDN - OVS Packet Flow Across Nodes

For pod to pod traffic on the same node, the traffic will go from pod 1 **veth0** to **br0**. NetworkPolicy rules are applied at **br0**. If a rule exists to permit the traffic flow, then the packet will flow to to pod 2 **veth0** and to pod 2 as depicted in 8. If the traffic is not permitted, the communication will be denied using the default rule and the packet will be dropped.

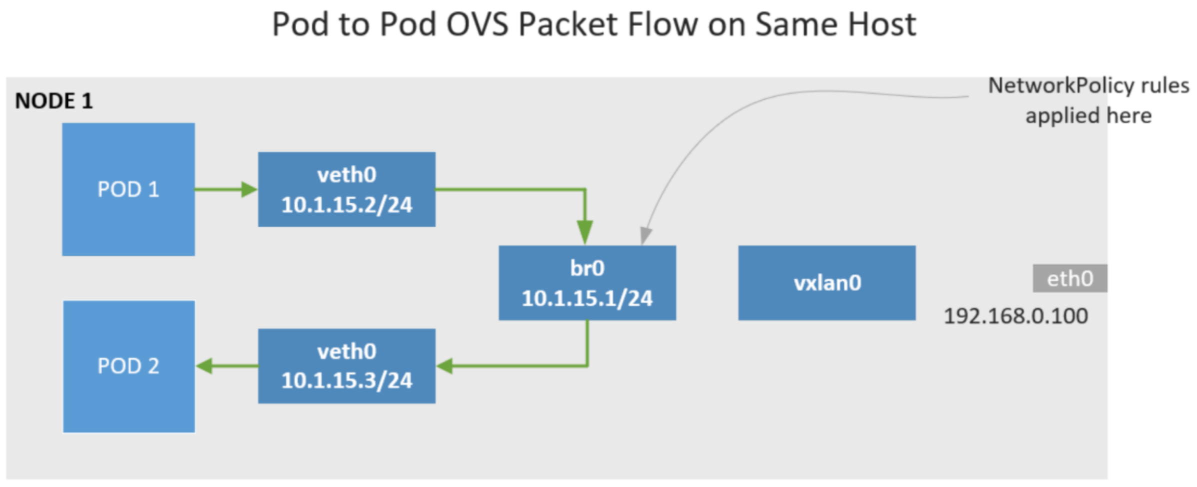


Figure 9 - OpenShift SDN - OVS Packet Flow Intra-Node

Ingress traffic coming in from the network external to the OpenShift environment to the router node is managed by the Ingress controller. It is also likely and recommended that all traffic coming into the OpenShift environment will first be inspected by physical edge firewalls, IDS/IPS, and so forth. This ingress traffic may also be configured for PCI DSS required TLS 1.2/1.3 cypher suite support of data encryption in transit. Figure 10 illustrates that traffic can be applied for ingress to explicitly permit traffic to certain pods. NetworkPolicy is in place to control packet flow between components of the application served by different pods. In this way, traffic can be limited to only what is needed for the proper function of the application.

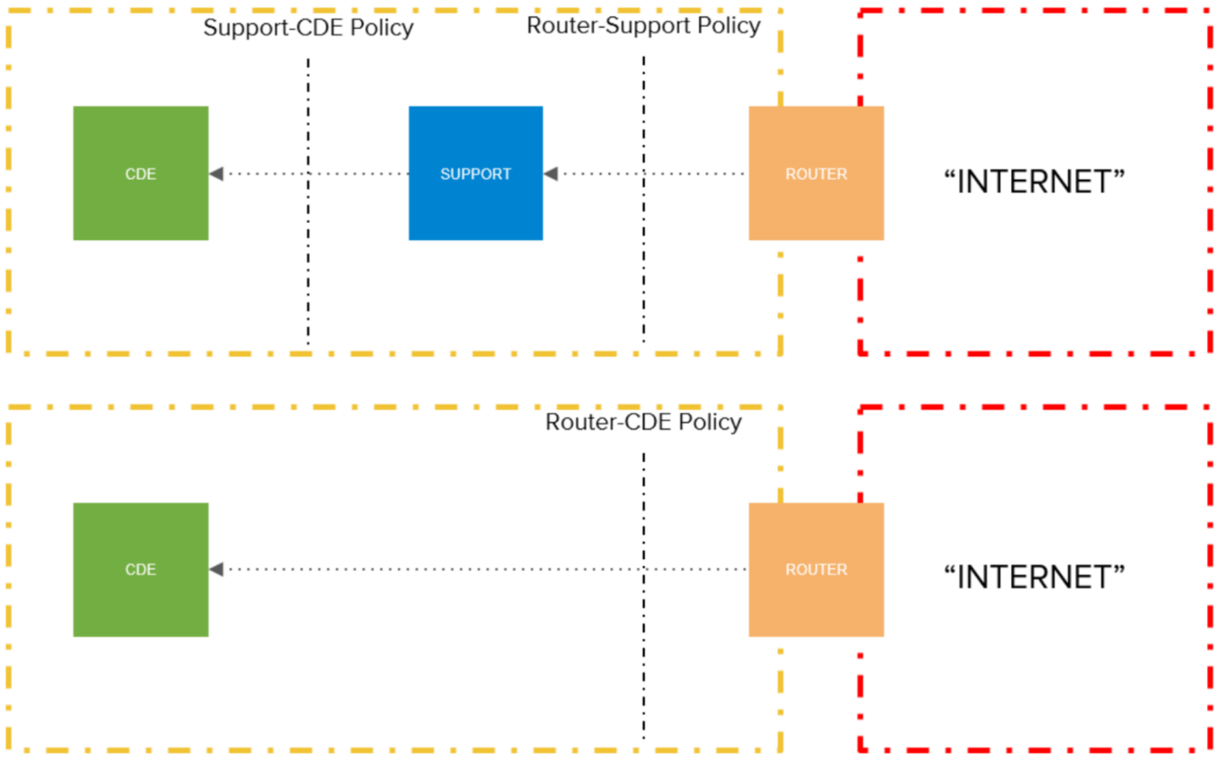


Figure 10 - Ingress Routing and Traffic Flow Control

Managing Egress

To add an additional layer of protection regarding the CDE systems in OpenShift pertaining to their access to external resources, the following two layers of security can be employed.

- Egress NetworkPolicy (firewall) – An egress network policy can be applied to a project where an application runs. The policy will act as a firewall, restricting the access to only the approved targeted off-cluster resources. As an example, the egress NetworkPolicy object can be used to ensure that an OpenShift project's pods can only access a specific Classless Inter-Domain Routing (CIDR) block or DNS name representing an external database outside of the cluster holding CHD or payment tokens.
- Static IPs for external traffic – When using the OpenShift SDN, each OpenShift project can receive an isolated network namespace. These namespaces can have IP addresses associated with them such that traffic leaving the OpenShift cluster appears to come from these IP addresses. The assignment of specified IP addresses to the controlled workloads allows for external (off-cluster) resources to whitelist the IP addresses for external access.

  By combining the two layers of security, access for the OpenShift workloads to external services or resources can be tightly controlled. Figure 11 illustrates the traffic flow control for egress utilizing these two methods.

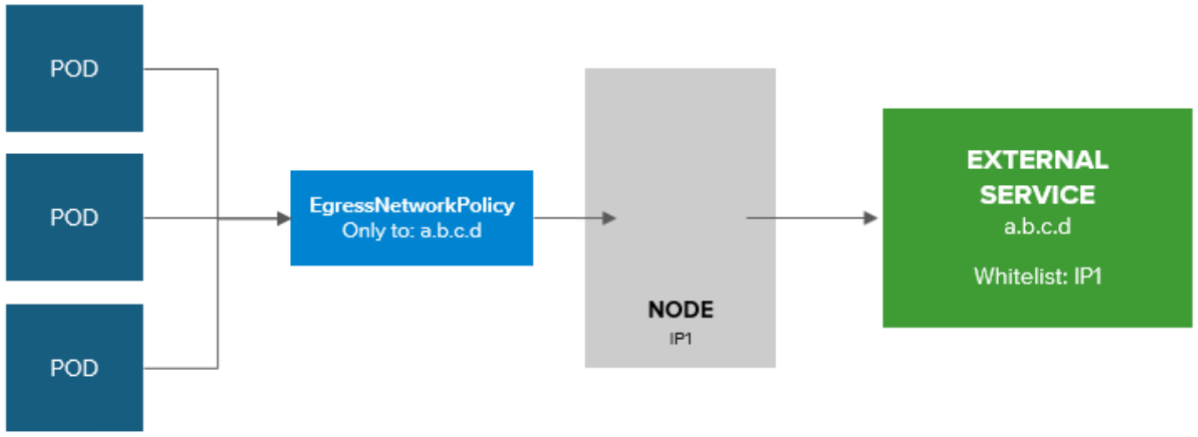


Figure 11 - Controlled Egress using NetworkPolicy Objects and Static IP

Using an egress router allows for the creation of identifiable services to send traffic to certain destinations, ensuring those external destinations treat traffic as though it were coming from a known source. This helps with security, because it allows for secure connections to external services, such as a database, so that only specific pods in a namespace can talk to a service (the egress router), which proxies the traffic to the external service. The egress router runs a service that redirects traffic to a specified remote server, using a private source IP address that is not used for anything else. This service allows pods to talk to servers that are set up to only allow access from whitelisted IP addresses. Figure 12 illustrates this type of controlled egress.

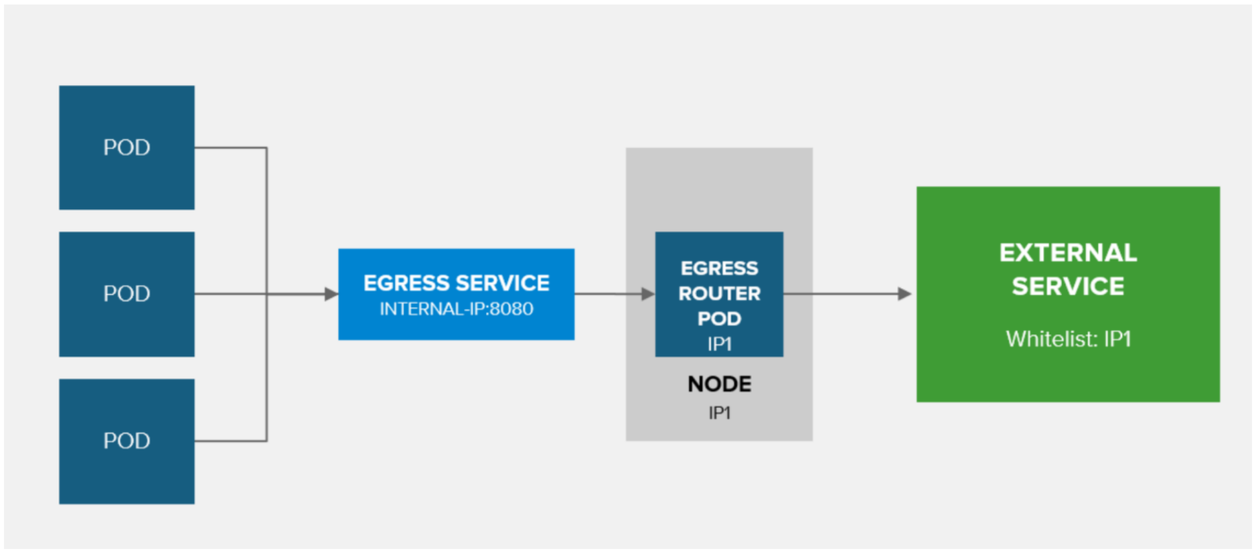


Figure 12 – Controlled Egress using Egress Service, Egress Router Pod, and External Service

Managing Container Registries –

It is recommended to block all container registries that are not explicitly known to the enterprise unless there are specific documented exceptions or reasons to allow access. For example, the enterprise should not allow access to anything but the Red Hat and/or enterprise private registries. Any additional third-party registries should be scrutinized with additional processes for testing images (scanning, tagging, etc.) that come from them before being admitted to the cluster.

OpenShift includes an integrated container registry that provides basic functionality supporting build and deployment automation within the cluster, tied into the OpenShift RBAC. Within the context of an organization needing to adhere to more stringent compliance requirements, such as PCI DSS 3.2.1, Red Hat Quay its optional Container Security Operator (CSO) is an additional product that provides a registry with capabilities for both RBAC and pod vulnerability scanning of applications and software in images and more. This CSO can be deployed to the OpenShift cluster, providing a view in the OpenShift console of known vulnerabilities in images deployed to the cluster from Quay.

Managing Users –

User (operators, developers, application administrators) access to OpenShift is provided through standard interfaces including the Web UI, CLI, and IDEs. These interfaces go through the authenticated and RBAC-controlled API. Users do not require system-level access to any of the OpenShift hosts, even for complicated application debugging and troubleshooting tasks.

VERIFICATION OF ADG FINDINGS

An actual construction of the design candidate for this ADG was created in a supplied Red Hat OCP lab and was used to support operation and analysis, and to perform the capture of screens and configuration snippets shown in this white paper.

The following sections represent in-scope requirements selected for testing along with the examined findings.

SEGMENTATION

The verification begins with review of capabilities to appropriately segment workloads within the OpenShift environment. Information Supplement: PCI SSC Cloud Computing Guidelines dated April 2018 Appendix E.7 states, "As containers can be used to achieve a similar level of execution, organizations may choose to use them to segment their PCI DSS scope in a manner similar to virtual machines. The Customer must validate that the container orchestration technology or solution offered by the Provider has all the features required to fully isolate containers." (Cloud Special Interest Group PCI SSC, 2018).

It is also noteworthy that the determination of actual compliance, in a real and not hypothetical payment card application using Red Hat CoreOS and OCP, is performed by the QSA responsible for the assessment (for merchants who meet or exceed the card transaction requirement for mandatory QSA reporting) and validation of sufficient isolation is based upon supplied artifacts and observations conducted by the QSA. Since segmentation is a best practice and not a requirement, selection of artifacts which verify isolation must adequately confirm Requirement 1 firewall controls and be satisfactory for any specific inquiries pertaining to scoping, virtualization and cloud guidance as provided in PCI SSC information supplements. This point of view is elaborated on in the following sections.

Background and Considerations

PCI DSS considers everything in scope until verified otherwise. Consequently, it is important to be able to demonstrate that the segmentation technique is sufficient to isolate CDE systems from out-of-scope systems.

One challenging statement in the PCI DSS segmentation guide is "In a flat network, all systems are in scope if any single system stores, processes, or transmits account data." The existence of separate network segments alone does not automatically create PCI DSS segmentation. Segmentation is achieved via purpose-built controls that specifically create and enforce separation and to prevent compromises originating from the out-of-scope network(s) from reaching CHD. The PCI DSS segmentation guide also states, "If network segmentation is in place and being used to reduce the scope of the PCI DSS assessment, the assessor must verify that the segmentation is adequate to reduce the scope of the assessment." (PCI SSC, 2017)

It will be necessary for the payment entity to sufficiently demonstrate that the logical constructs in place are sufficient to provide isolation such that no component of out-of-scope systems can poison the CDE in any way including, but not limited to, the spread of malware from a compromised system, unauthorized east-west access, and capture of packets transmitted over a common transit network. Depending on the size and complexity of the organization, out-of-scope systems being included in scope due to lack of adequate segmentation may result in increased cost. The same security measures to address compliance requirements would need to be applied to all systems, users, networks, storage, and so forth, regardless of how the payment entity categorizes the system. All systems, devices, users, networks, storage, and so forth would be in scope for assessment, thus increasing the cost of assessment.

Consequently, many organizations will opt to build out entirely separate infrastructure to support their CDE systems physically isolated from their out-of-scope or general support systems. This is not only seen with traditional physical server infrastructures, but also with virtualized infrastructures where payment entities are building separate virtualization infrastructure stacks to service each type of workload independently. The advent of SDN solutions has eased some of the restrictiveness, which can provide support for improved consolidation and efficiency while maintaining workload isolation. Despite the logical segmentation capabilities afforded by SDN, organizations are more often opting to, at a minimum, continue to isolate CDE systems to separate compute nodes. In this case, consolidation of resources would be limited to the sharing of common management and control planes, where the data plane would continue to be isolated and unique to each security category. The management and control planes would be scoped according to their proximity and ability to influence or impact the CDE systems and would be considered connected-to and security-impacting systems. In most cases, the common management and control plane are in scope; consequently, there should be careful consideration for placement of these components within the network and on compute nodes as well. Ideally, these components would be segmented to their own management network and likewise their own compute nodes.

Use of OCP to satisfy the previous considerations does allow for consolidated infrastructure if the requirement for firewall isolation can be demonstrated to the PCI QSA via verification of access control rules restricting network access to eliminate all out-of-scope system access to the CDE and to vise-versa. In the following sections on segmentation testing, tangible examples of QSA verification methods are described.

Regardless of logical methods to isolate virtual machines, any compute node that hosts CDE systems would be considered, by extension, in scope for assessment.

Segmentation Testing

Test that segmentation controls prevent communication from out-of-scope systems to CDE systems.

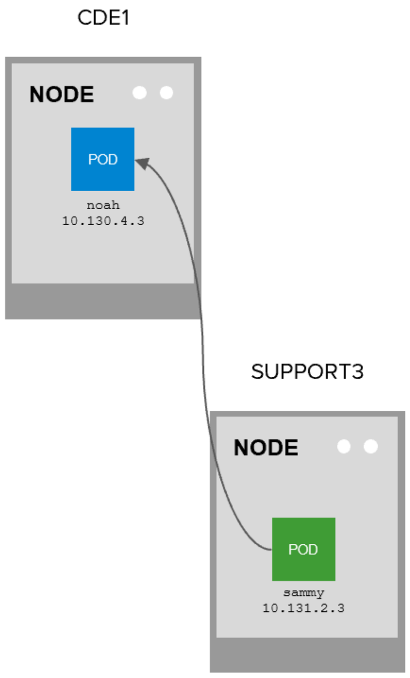
Coalfire found that NetworkPolicy rules allow for restriction of traffic flowing into applications. NetworkPolicy ruless essentially work like a virtual firewall. This allows for micro-segmentation of workloads within the OpenShift environment to control network traffic between pods in the OpenShift environment. To isolate one or more pods in a project, NetworkPolicy ruless can be created to indicate the allowed incoming connections. Pods that do not have NetworkPolicy ruless pointing to them are fully accessible, whereas pods that have one or more NetworkPolicy ruless pointing to them are isolated. Isolated pods only accept connections that are accepted by at least one of their NetworkPolicy rules.

A default deny-all policy was created such that even pods in the same project are unable communicate with each other on the network.

```
# deny everything into the CDE1 project
# even resources in project can't reach each other
---
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
 name: deny-by-default
spec:
 podSelector:
 ingress: []
```

A policy was created to authorize specific traffic between pods.

```
# allow sammy (support1-app1) to reach the noah
(cde1-app1)
---
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
 name: sammy-to-noah
spec:
 # destination is noah-labeled pods
 podSelector:
   matchLabels:
     role: noah
 # source is sammy specifically in support1
(backyard)
 # in 3.10 can only allow entire namespace access
 ingress:
   - ports:
     # destination is tcp 8080
     - protocol: TCP
       port: 8080
     from:
       - namespaceSelector:
           matchLabels:
             location: backyard
```



This explicitly permits traffic between pods with the label sammy to pods with the label noah with protocol TCP and port 8080.

The results of the NetworkPolicy ruless show that Support3 flow rule tags packets from sammy source pod IP (because it has the correct labels) with the VNID of the Support1 Namespace.

```
cookie=0x0, duration=1033474.948s, table=20, n_packets=2505533,
n_bytes=439815806, priority=100,ip,in_port=4,nw_src=10.131.2.3
actions=load:0xce981f->NXM_NX_REG0[],goto_table:21
```

CDE1 flow rule tags packets destined for noah pod IP (because it has the correct labels) with the VNID of the CDE1 Namespace.

```
cookie=0x0, duration=1033587.577s, table=70, n_packets=2491997,
n_bytes=721525199, priority=100,ip,nw_dst=10.130.4.3
actions=load:0x334953->NXM_NX_REG1[],load:0x4-
>NXM_NX_REG2[],goto_table:80
```

CDE1 flow rule allows packets destined for noah pod IP (because it has the correct labels) with the VNID of the CDE1 Namespace and the VNID of the Support1 Namespace.

```
cookie=0x0, duration=1246.896s, table=80, n_packets=0, n_bytes=0,
priority=150,tcp,reg0=0xce981f,reg1=0x334953,nw_dst=10.130.4.3,tp_dst=8
080 actions=output:NXM_NX_REG2[]
```

The SDN implementation is responsible for tracking all NetworkPolicy rules and creating OVS flow rules on the nodes and for the POD IPs as required.

Packet flows are only possible when an explicit flow rule exists. Otherwise the behavior is to deny-by-default (because of the deny all NetworkPolicy rule).

Across projects, the policy can allow everything or nothing. Inside a project, one can be granular in how the pods can communicate with each other.

REQUIREMENT 1

Install and maintain a firewall configuration to protect cardholder data.

Requirement 1 is primarily concerned with traditional edge protections between the Internet or "untrusted networks" and internal networks. It is recommended that the OpenShift environment be placed in an internal controlled network that is protected with traditional edge protections provided by third-party solutions. As such, Coalfire determined that most, if not all, of these requirements were not pertinent to OpenShift's capabilities. However, assessors often look at implementation of firewalls and routers on internal networks used to isolate or segment workloads as a method of reducing assessment scope in pursuit of the best practice of network segmentation to reduce scope. With this in mind, many of the requirements may apply to the internal network elements that perform segmentation, including the SDN elements provided by OpenShift.

Where firewalls are required in the review conducted for this ADG, they were considered to be present notionally, and would be supplied outside of the context of Red Hat CoreOS and OCP.

1.1 Establish and implement firewall and router configuration standards that include the following:

In addition to standards that address traditional firewalls and routers, the payment entity's firewall and router standards should also address the use of SDN constructs to provide micro-segmentation capabilities and/or isolate workloads within the internal network. Much like traditional network hardware, these technologies can be used to efficiently enable segmentation and isolation of workloads in support of compliance objectives as well as provide additional layers of protection between a DMZ and internal secure networks.

1.1.1 A formal process for approving and testing all network connections and changes to the firewall and router configurations.

Included with procedures for testing and evaluating traditional edge protections, the payment entity should additionally provide documented procedures for testing the efficacy of NetworkPolicy ruless to provide the entity's designed segmentation of workloads, and clearly note how they relate to firewalls preforming enforcement of security rules. These procedures would be used for testing approved network connections as well as changes to firewall and router configurations. This should include changes to NetworkPolicy objects, as well as modifications to OpenShift ingress and egress routing methods in support of workloads hosted in OpenShift. The payment entity should be able to demonstrate for assessment purposes that the network controls are sufficient to prevent unauthorized access from non-CDE systems to CDE systems.

1.1.2 Current network diagram that identifies all connections between the cardholder data environment and other networks, including any wireless networks.

In addition to the physical network infrastructure supporting OpenShift, the payment entity should be prepared to include network topography diagrams that include virtual network connections within the OpenShift environment that illustrate virtual network connectivity between containerized workloads. Where workloads are segmented, logical topography diagrams should include the control mechanisms that isolate and/or restrict communication between workloads.

1.1.3 Current diagrams that show all cardholder data flows across systems and networks.

The payment entity would also be responsible for illustrating the flow of CHD across the systems and networks. These illustrations should be able to identify the methods and techniques for isolating workloads in support of segmentation and scope reduction efforts to maintain a separate and isolated CDE systems segment within the OpenShift environment.

Diagrams typically provided include data flows performed by container (and conventional virtualized machine) CDE systems within security boundaries for CDE and C-T/SIS assets, as well as dataflows from end-users (in the case of web commerce systems, or other on-line activities), operators and automated processes within the payment system. Payment processor dataflows via dedicated circuits and over the internet are also a required part of these diagrams. These diagrams should ideally be part of a company-wide portfolio of documentation which depicts PCI DSS in-scope and other corporate LAN, out-of-scope systems as a whole.

1.1.4 Requirement for a firewall at each Internet connection and between any demilitarized zone (DMZ) and the internal network zone.

It is recommended to place the OpenShift environment on the internal network protected by the payment entity's placement of firewalls at each Internet connection and between any DMZ and the internal network zone.

1.1.5 Description of groups, roles, and responsibilities for management of network components.

In addition to groups, roles, and responsibilities defined for traditional network components, the payment entity should also identify and describe the groups, roles, and responsibilities for management of the OpenShift SDN.

1.1.6 Documentation of business justification and approval for use of all services, protocols, and ports allowed, including documentation of security features implemented for those protocols considered to be insecure.

The payment entity will be responsible for providing documented justification for the use of services, protocols, and ports allowed. This includes the use of services, protocols, and ports relative to the workloads deployed in OpenShift as well as the services, protocols, and ports used by the OpenShift environment. Red Hat provides a listing of services, protocols, and ports that are required for the proper functioning of OpenShift that is readily available from their customer portal.

Excluding payment entity deployed workloads in the OpenShift environment, in general, services, protocols, and ports that are insecure are not commonly in use by OpenShift. The payment entity should take care to avoid the use of insecure ports, protocols, and services such as FTP, Telnet, POP3, IMAP, SNMP v1 and v2, and other such protocols where data is transmitted in the clear without sufficiently strong encryption to prevent compromise. Some services like DNS, LDAP and NTP are not commonly encrypted, but may be allowable depending on the degree that these services could be used as an attack vector. It is of utmost importance that the payment entity avoids clear, unencrypted transmission or storage of authentication credentials that could be used by an attacker to gain unauthorized access to the system. OpenShift is configured to use secure LDAP. Likewise, CHD should not be transmitted over the network in the clear, especially where the transport zone is shared with systems other than CDE systems.

1.1.7 Requirement to review firewall and router rule sets at least every six months.

In addition to the inspection of edge firewall and router rule sets and where the payment entity is using SDN to support segmentation, the payment entity should also include inspection of any internal firewall and router rulesets defined within the virtualized or containerized environment; this includes those rulesets defined and enforced within the OpenShift environment.

1.2 Build firewall and router configurations that restrict connections between untrusted networks and any system component in the cardholder data environment. Note: *An "untrusted network" is any network that is external to the networks belonging to the entity under review, and/or which is out of the entity's ability to control or manage.*

It is recommended to implement OpenShift within the controlled internal network. Network protection for the OpenShift environment would be provided by the payment entity's edge, DMZ and C-T/SIS firewall or ACL-based security solution.

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

This protection is primarily provided by the payment entity's implementation of, external to OpenShift, firewalls and routers between the Internet edge and the internal network. Egress traffic from OpenShift can be made identifiable to allow for proper application of outbound policy at the network's edge. This allows for egress traffic for CDE systems hosted within OpenShift to be distinguishable from support system traffic or out-of-scope system traffic.

Additionally, OpenShift SDN NetworkPolicy objects can be used to control the flow of traffic within the OpenShift environment, including the ability to establish policies by workload to explicitly deny all traffic and allow traffic by exception.

1.2.2 Secure and synchronize router configuration files.

While primarily applicable to the organizations edge protections, access to the OpenShift SDN configuration can be controlled through RBAC. All changes made to the SDN configuration are immediate and synchronized throughout the OpenShift environment.

1.2.3 Install perimeter firewalls between all wireless networks and the cardholder data environment, and configure these firewalls to deny or, if traffic is necessary for business purposes, permit only authorized traffic between the wireless environment and the cardholder data environment.

This may not be applicable to OpenShift or the contained workloads. Coalfire recommends implementing OpenShift on a wired network within the secure confines of the payment entity's facilities. Where wireless networks are utilized by the payment entity, the payment entity must implement a perimeter firewall between any wireless network in use by the enterprise and the CDE on the wired network.

1.3 Prohibit direct public access between the Internet and any system component in the cardholder data environment.

The expectation is that the choke router at the Internet, the DMZ router and firewall, and/or the perimeter firewalls and routers will provide a layer of protection between the Internet and the internal systems, including the OpenShift environment. This allows for adequate inspection and filtering of requests from the internet to determine the legitimacy of the request and to properly direct the traffic and ensure that it meets security requirements before passing the traffic to the internal workloads and/or DMZ-hosted workloads. This includes the implementation of a DMZ to limit inbound traffic to only system components that provide authorized publicly accessible services, protocols, and ports (**1.3.1**) and limiting inbound Internet traffic to IP addresses within the DMZ (**1.3.2**). To a limited degree, OpenShift ingress routing and the NetworkPolicy objects can support restriction of inbound traffic to pods that are designated for external access. In this way, pods that are purely internal can be isolated from any access external to the OpenShift environment. In a similar way, authorization of outbound traffic can be supported with an architecture within OpenShift using egress policy objects or an egress service in coordination with IP mappings to direct traffic to be handled further by an external firewall (**1.3.4**).

Likewise, the external network security devices should provide required anti-spoofing measures to detect and block forged source IP addresses from entering the network (**1.3.3**). This can be provided by a third-party stateful firewall (**1.3.5**) with anti-spoofing features and is external to OpenShift.

Systems can be further segmented (**1.3.6**) to provide further isolation such that internal systems are placed on an internal network zone and segregated from the DMZ or other untrusted networks. Where some pods within the OpenShift environment can be implemented to allow ingress or egress, others can be strictly enabled for internal communication only, including internal and designated connections to data sources, such as databases.

Protections against disclosure of private IP addresses (**1.3.7**) should be handled by solutions external to OpenShift such as edge firewalls or routers, or DMZ routers or firewalls. These devices are more equipped to provide services such as Network Address Translation (NAT). With the OpenShift SDN, workloads are identified by DNS name and IP addresses, which are dynamically assigned to pods based on namespace designations and are obfuscated outside of the OpenShift environment. OpenShift ingress routers direct traffic to targets based on mappings, much like a load balancer.

1.4 Install personal firewall software or equivalent functionality on any portable computing devices (including company and/or employee-owned) that connect to the Internet when outside the network (for example, laptops used by employees), and which are also used to access the CDE. Firewall (or equivalent) configurations include:

- **Specific configuration settings are defined.**
- **Personal firewall (or equivalent functionality) is actively running.**
- **Personal firewall (or equivalent functionality) is not alterable by users of the portable computing devices**

It is not recommended to install OpenShift on personal computing devices. The organization will want to ensure that endpoint devices that are being used by administrators and users of the OpenShift environment are sufficiently protected to meet this requirement.

1.5 Ensure that security policies and operational procedures for managing firewalls are documented, in use, and known to all affected parties.

The organization will want to ensure that they include in security policies and operational procedures for managing the OpenShift SDN and OpenShift NetworkPolicy objects within the OpenShift environment.

These procedures should be part of the organization’s active documentation and understood by those designated as responsible parties for managing these network components.

REQUIREMENT 2

Do not use vendor-supplied defaults for system passwords and other security parameters

Remove or disable unnecessary default accounts and authenticators. This includes default user accounts, administrator accounts, root accounts, and SNMP community strings. If an account is necessary to the function of the OS, the application, or the system, there will need to be a justification for the account’s existence. At the very least, the authenticators or passwords should be uniquely set during the initial setup and should be protected from unauthorized access. When these accounts are required for normal operation of a system, they should be monitored for abnormal or anomalous activities that are likely indicators of compromise or unauthorized access. Likewise, generic accounts like root or system:admin are only used interactively in cases of emergency recovery and audited for approved usage. Normal administrative and user access should always take place with named user accounts, which personally identifies the user for whom the account is assigned.

2.1 Always change vendor-supplied defaults and remove or disable unnecessary accounts before installing a system on the network. This applies to ALL default passwords, including, but not limited to those used by operating systems, software that provides security services, application and system accounts, point-of-sale (POS) terminals, payment applications, Simple Network Management Protocol (SNMP) community strings, etc.).

No vendor-provided default passwords are provided or in use for OpenShift or Red Hat CoreOS. Authenticators are uniquely created at the time that OpenShift is deployed, including a kubeadmin account, which has a per-system unique, randomly generated password. This kubeadmin account is traditionally removed, after the external IDP is configured.

The initial cluster administrator account kubeadmin, present at the onset of installation, should be deprecated after the creation of working cluster administrators which are then subsequently used for OpenShift administration thereafter. The only users that exist on an Red Hat CoreOS OpenShift node are *root* and *core*. The core user is a member of the *wheel* group, which gives it permission to use sudo for running privileged commands. Adding additional users at the node level is highly discouraged. By default, the only way to access a shell on a node in an OpenShift cluster is via the CLI command oc debug node/<node>. It is important to note that this provides a shell logged in as root on the node. This is only available to users with the cluster-admin role. Users with the cluster-admin role should be limited as much as possible.

The host’s root account is necessary for OS operations as well as some OpenShift host interactions. It has no default password and is not SSH-enabled. Default authenticators are protected such that no other users to the system or underlying OS can access the protected folders or files.

2.1.1 is not applicable to OpenShift as OpenShift should not be deployed in a wireless environment, nor does it provide wireless capabilities. Where the payment entity uses wireless environments for their CDEs, the payment entity will be responsible for meeting 2.1.1 requirements relative to wireless vendor defaults and security settings.

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. Sources of industry-accepted system hardening standards may include, but are not limited to:

- **Center for Internet Security (CIS)**
- **International Organization for Standardization (ISO)**
- **SysAdmin Audit Network Security (SANS) Institute**
- **National Institute of Standards and Technology (NIST)**

The payment entity is responsible for developing configuration standards and adhering to industry-accepted system hardening standards. This responsibility applies to the OpenShift environment, including OpenShift, container security, and the underlying operating environment upon which OpenShift is deployed. Red Hat provides guidance with best practices concerning Red Hat CoreOS, OpenShift, and the container environment deployments. The Red Hat OpenShift Container Security Guide is a good resource for properly deploying and securing the platform and the workloads.

OpenShift Compliance Operator

The optional Compliance operator is available starting with OpenShift 4.6 (outside of the scope of this ADG). The Compliance Operator lets OpenShift Container Platform administrators audit the desired compliance state of a cluster and provides them with an overview of gaps and ways to remediate them. The Compliance Operator assesses compliance of both the Kubernetes API resources of OpenShift Container Platform, as well as the nodes running the cluster. The Compliance Operator uses OpenSCAP, a NIST-certified tool, to scan and enforce security policies provided by the content.

OpenShift runs on Red Hat CoreOS and makes use of security capabilities of the host OS. The following are features of Red Hat CoreOS that make containers secure on the platform.

- Linux kernel namespaces enable creating an abstraction of a particular global system resource to make it appear as a separate instance to processes within a namespace. Consequently, several containers can use the same resource simultaneously without creating conflict.
- SELinux provides an additional layer of security to keep containers isolated from each other and from the host. SELinux allows administrators to enforce MAC for every user, application, process, and file.
- CGroups (control groups) limit, account for, and isolate the resource usage (CPU, memory, disk I/O, network, etc.) of a collection of processes. CGroups are used to ensure that containers on the same host are not impacted by each other.
- Secure computing mode (seccomp) profiles can be associated with a container to restrict available system calls.
- Deploying containers using a CoreOS reduces the surface area of attack by minimizing the host environment and tuning it specifically for containers.

These capabilities protect the workloads running in containers such that different workloads do not have access to keys, identity tokens, or other sensitive information used by other containers in a cluster. The containers can only see what the container is authorized, by design, to see and cannot see or access another container’s information. The container security capabilities of OpenShift on Red Hat CoreOS are in alignment with and in support of the PCI SSC Cloud Computing Guidelines - Information Supplement Appendix E.7.

The OpenShift environment provides network and administrative isolation capabilities between containers hosting different workloads based on the container-specific network interface and SDN. OpenShift administrators and project administrators can establish NetworkPolicy rules that control the flow of information between containers per the payment entity's design and in support of PCI SSC recommendations for segmentation.

Changes to NetworkPolicy rules including creation, modification, and/or deletion are logged and can be reviewed according to the payment entity's change control policies. It is recommended that OpenShift Container Platform administrators configure RBAC such that only specific, known users may modify NetworkPolicy objects for in-scope namespaces/projects.

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. (For example, web servers, database servers, and DNS should be implemented on separate servers.) Note: Where virtualization technologies are in use, implement only one primary function per virtual system component.

As a virtualization technology, OpenShift provides the means to separate application workloads within the OpenShift cluster onto separate hosts and containers within the cluster. A significant and primary benefit of using containers helps to eliminate the vulnerability which this control addresses: multiple functions on a single virtual machine/bare-metal server. Proper container and pod design is required to support this control.

2.2.2 Enable only necessary services, protocols, daemons, etc., as required for the function of the system.

The payment entity can harden the OS, pods and containers to limit the services, protocols, daemons, etc., as required for the function of the applications in the CDE. Although RHEL CoreOS only contains the services, protocols and daemons needed to run OpenShift, Red Hat provides guidance pertaining to additional platform hardening options in their documentation.

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

OpenShift does not require insecure services, protocols, or daemons for proper functioning. It is recommended to avoid the use of insecure services, protocols, or daemons either with the platform components or with the workloads that are deployed with OpenShift. See 1.1.6 for details.

2.2.4 Configure system security parameters to prevent misuse.

The payment entity's OpenShift administrators and users should be aware of the configuration parameters that are in use for securing the OpenShift environment.

2.2.5 Remove all unnecessary functionality, such as scripts, drivers, features, subsystems, file systems, and unnecessary web servers.

The payment entity will deploy Red Hat CoreOS which does not include any extra unnecessary functionality.

2.3 Encrypt all non-console administrative access using strong cryptography.

The only way to access the OpenShift Control Plane or other OpenShift cluster hosts for non-console administrative access is through the OpenShift CLI or SSH. Administrative access to the OpenShift cluster via the CLI requires an account with the cluster-admin role. The only host user that is SSH-enabled is the "core" user, but it has no SSH keys by default, and is not enabled for password authentication. Logging into the Red Hat CoreOS host with SSH is discouraged as all administrative functions can be managed through the CLI by a user with the OpenShift cluster admin role. Telnet is not enabled. Administrative access to the API or the Web User Interface occurs over TLS 1.2/1.3.

2.4 Maintain an inventory of system components that are in scope for PCI DSS.

The payment entity will be responsible for maintaining an inventory of system components, including hardware and software, that are in use for their OpenShift environment.

2.5 Ensure that security policies and operational procedures for managing vendor defaults and other security parameters are documented, in use, and known to all affected parties.

The payment entity will be responsible for the documentation and dissemination of the security policies and operational procedures pertaining to their OpenShift deployment and verifying that the policies and procedure are in use and being followed.

REQUIREMENT 3

Protect stored cardholder data

This requirement pertains to methods used within the application software to ensure that data at rest encryption (DARE) is used pervasively wherever PCI DSS CHD (primary account numbers / PAN and sensitive authentication data / SAD) might be stored onto written, recorded, rotating disk or solid-state disk media. Largely the responsibility of the developer, Red Hat CoreOS provides for means of full disk encryption which can optionally use its' FIPS 140 certified encryption libraries and built-in functionality to access that data. Disk encryption is supported is either with TPM2 or a Tang server. Red Hat CoreOS disk encryption is FIPS compliant if FIPS mode is enabled.

A subset of requirement 3 applies to Red Hat CoreOS and OpenShift implementations.

3.2 Do not store sensitive authentication data after authorization (even if encrypted). If sensitive authentication data is received, render all data unrecoverable upon completion of the authorization process.

Proper design of the application by the payment entity can accommodate this requirement as a processing mandate, restricting in-memory process for this data and taking care not to write to file storage from within the container or pod.

3.4 Render PAN unreadable anywhere it is stored (including on portable digital media, backup media, and in logs) by using any of the following approaches:

Similar approaches to 3.2 for properly coding the handling of PAN when stored are mandated by this control. This is the responsibility of the code developer and for bespoke applications using OCP this should become a development standard. These requirements were not specifically reviewed in our ADG lab activities and were notionally represented.

3.6 Fully document and implement all key-management processes and procedures for cryptographic keys used for encryption of cardholder data, including the following: …

3.6.1 Generation of strong cryptographic keys

Payment card QSAs will look for support of this control family of requirements and the underlying support for use of "strong cryptography" supported by Red Hat CoreOS FIPS 140 cryptographic libraries and the cryptographic libraries available in the RHEL Universal Base Image when reviewing DARE schemes for the PCI entity. PCI DSS reference material qualifies specific crypto-cipher suites and key strategies to satisfy these requirements.

REQUIREMENT 4

Encrypt transmission of cardholder data across open, public networks

There are multiple options for encrypting ingress traffic to applications, including passthrough encryption and re-encrypt.

Application developers are responsible for encrypting service to service traffic. OpenShift includes options to support developers, including the Service CA. And OpenShift Service Mesh is available to encrypt pod to pod traffic within an OpenShift cluster.

When booted in FIPS mode, RHEL CoreOS supports FIPS 140 validated libraries available in RHEL UBI as primitives to container developers, to provide TLS 1.2 cryptographic measures and supply data in motion encryption (DIME) to satisfy this control requirement. As the requirement states, “Sensitive information must be encrypted during transmission over networks that are easily accessed by malicious individuals.”

4.1 Use strong cryptography and security protocols to safeguard sensitive cardholder data during transmission over open, public networks, including the following:
• Only trusted keys and certificates are accepted.
• The protocol in use only supports secure versions or configurations.
• The encryption strength is appropriate for the encryption methodology in use.

Secure coding practices and organizational procedures must be in force for the OCP environment to satisfy this control family. Standards for minimum keys and certificate issuance need to be created and observed during the development process with special attention to make TLS 1.2 the minimum security for HTTPS and other uses of certificate-based crypto for the entity.

For built-in API and web services, containers must support the “strong encryption” mandate by selection of appropriate cipher suites and implementation of client and server certificates. It is also a strong recommendation that where practical, TLS is used for internal transactions between systems components and ideally throughout the payment card application. OpenShift Service Mesh can be used to automatically encrypt pod-to-pod traffic for in-scope services.

OpenShift includes the ability to configure cipher suites using the tlsSecurityProfile parameter. This object is used by operators, including the Ingress, API Server and Authentication operators, to apply TLS settings.

REQUIREMENT 5

Protect all systems against malware and regularly update anti-virus software or programs

Security techniques that are commonly used by traditional IT infrastructures have limited functionality in containerized infrastructures. The payment entity must consider the availability of optional security solutions for vulnerability detection, malware detection, intrusion detection, intrusion prevention, and network flow analysis. A key aspect to successful security of container environments is identifying and understanding the opportunities or gateways for detection.

If applicable, containerized anti-virus software exists, it should be deployed to the Red Hat CoreOS hosts at a minimum according to this requirement. However, given that the user space on Red Hat CoreOS is read-only, there may be limited value to deploying anti-virus software. Alternatively, the **OpenShift File Integrity Operator** can be deployed to monitor file system integrity on the host.

The optional File Integrity Operator can be deployed to continually run file integrity checks on the cluster nodes. It deploys a daemon set that initializes and runs privileged advanced intrusion detection environment (AIDE) containers on each node, providing a status object with a log of files that are modified during the initial run of the daemon set pods.

As applications are deployed from container images, the payment entity should regularly scan image repositories for malware or malicious software. The payment entity may choose, as part of the systems development lifecycle (SDLC), to also scan images prior to placement in the registry and prior to being enabled for deployment to ensure that the images are free of malware or malicious software. Red Hat certified images, monitored for newly discovered vulnerabilities and updated as necessary, are supported by Red Hat and/or other third-party software owners. Red Hat advisories alert administrators to newly discovered issues or vulnerabilities and direct the administrator to use updated images. OpenShift provides a pluggable API to its CVE data to support multiple vulnerability scanners.

Red Hat Quay is an optional container registry that can be leveraged with built in vulnerability scanning capability to scan stored applications and images for known vulnerabilities.

REQUIREMENT 6

Develop and maintain secure systems and applications

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 vulnerabilities.

The payment entity should establish a process to identify security vulnerabilities, using reputable sources for security vulnerability information that includes resources that track vulnerabilities that commonly impact container environments, Red Hat OpenShift, and Red Hat CoreOS. Red Hat is dedicated to addressing vulnerabilities and making security patches and updates available to their customers. Updates to OpenShift, including Red Hat CoreOS, are routinely made available for deployment to maintain the latest version of OpenShift.

For images that the payment entity desires to use, there are tools that can be used to scan and track contents of downloaded and deployed container images. It is important when using public container registries to use trusted sources.

The process for consuming security updates for containers is important and different than that of traditional application environments. It is recommended to use immutable containers which are containers that will never be changed while running. This is important because imutable containers helps protect the container from compromise as a result of code being injected or other such attacks. With traditional application architectures, updates are applied directly to the application where binaries are replaced; whereas, with immutable containers, there is a rebuild and redeploy process to update the application. The payment entity should not patch running containers but should rebuild and re-deploy them.

Red Hat provides a trusted repository for images where Red Hat regularly fixes known vulnerabilities per its errata policy in the platform components or layers. Red Hat certified images are compatible across the Red Hat CoreOS platforms, from bare metal to cloud. Finally, Red Hat-certified images are supported by Red Hat or Red Hat's partners. While the list of known vulnerabilities is evolving as new vulnerabilities are discovered all the time, the payment entity must track the contents of their deployed container images, as well as newly downloaded images, over time. The payment entity can use Red Hat Security Advisories (RHSAs) to alert them of any newly discovered issues in Red Hat certified container images and direct them to the updated image ready for deployment.

Red Hat uses a health index for security risk with containers provided through the Red Hat Container Catalog.

6.2 Ensure that all system components and software are protected from known vulnerabilities by installing applicable vendor-supplied security patches. Install critical security patches within one month of release.

As Red Hat routinely manages the images that are provided in their trusted repository, the payment entity will be able to address vulnerabilities in a timely manner.

6.4 Follow change control processes and procedures for all changes to system components. The processes must include the following:

The payment entity should update their documented change control processes and procedures to include procedures that are unique to the implementation of change in an OpenShift environment as opposed to traditional environments. Likewise, these will include changes that the payment entity may desire to make to the workloads that the payment entity deploys and orchestrates within the OpenShift environment.

As part of the change control processes and procedures, the payment entity must separate development/test environments from production environments and enforce the separation with access controls (**6.4.1**). OpenShift can be implemented in a way to allow for separation of development/test environments from production environments including the use of RBAC, projects and network policy rules to enforce separation. Access controls can be applied to projects where separate projects can be created for each of development and test environments distinct from the production environments. In addition to access controls to establish isolation and prevent unauthorized changes from being made to production, OpenShift SDN can provide network isolation for these different namespaces as well. Finally, it is recommended to sequester development and test environments to separate hosts.

Access controls for administration and management of OpenShift as well as on a project basis can be established to support separation of duties; different levels of access can be granted to development and test projects and users distinct from production projects and users (**6.4.2**).

The remainder of requirement 6 is pertinent to the payment entity’s development practices related to secure development of in-house custom applications.

REQUIREMENT 7

Restrict access to cardholder data by business need to know

Combined with third-party identity and authentication providers (See requirement 8), OpenShift provides the means to enable granular access control through RBAC. OpenShift provides authorization mechanisms to control access at the cluster level, as well as at the project level.

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

The payment entity will need to address policy for access controls that incorporate 7.1.1 through 7.1.4 as it pertains to OpenShift. This includes including definition of access needs and privileged assignments for each role for access to the OpenShift environment and components. It will be important for the security of the OpenShift environment to restrict access to privileged user IDs to the least privilege necessary to perform the job responsibilities that are assigned. Access should be assigned based on the personnel’s job classification and function as a justification for access. Finally, granted access should include documented approval for such access by authorized parties for all access, including listing of specific privileges approved.

7.2 Establish an access control system(s) for systems components that restrict access based on a user's need to know and is set to "deny all" unless specifically allowed. This access control system(s) must include the following:

OpenShift can be configured to support the payment entity's standards for access control based on minimum access requirements. RBAC can be used to establish authorization with a granular level of control for interaction with and/or administration of OpenShift and the underlying operating system, Red Hat CoreOS. No initial access is granted, except for the access that is established upon initial setup.

It is recommended to use a strict identity and access control mechanism. Identity and initial authentication are provided by a third-party solution, external to OpenShift. The OpenShift Control Plane includes a built-in OAuth server. Users obtain OAuth tokens to authenticate themselves to the API. An administrator can configure OAuth to authenticate against an identity provider. OpenShift has several available options for integration with external identity and authentication providers such as LDAP, Active Directory, GitHub, or Google, and others. The environment used LDAP with an external identity provider.

(**7.2.1**) Coverage for authentication and authorization includes support for all components of OpenShift from the underlying OS up to the container itself. Most users will access OpenShift through normal mechanisms, either the Web UI, the CLI or the API. The payment entity will be required to establish authentication and authorization parameters for the workloads hosted by OpenShift.

The authentication layer identifies the user associated with requests to OpenShift's API. The authorization layer then uses information about the requesting user to determine if the request should be allowed. RBAC objects (**7.2.2**) determine whether a user can perform a given action. This allows platform administrators to use the cluster roles and bindings to control who has various access levels to OpenShift itself and all projects. Authorization is managed using rules, roles, and bindings.

Rules are a set of permitted verbs (actions: get, list, create, update, delete, deletecollection, or watch) on a set of objects (containers and images, pods and services, projects and users, builds and image streams, deployments, routes, and templates). For example, whether someone or something can create pods.

Roles are a collection of rules. Users and groups can be associated with, or bound to, multiple roles at the same time, allowing for granular control over authorization rights granted to a user.

Bindings are associations between users and/or groups with a role.

The relationship between cluster rules, local roles, cluster role bindings, local role bindings, users, groups, and service accounts are illustrated in the following figure.

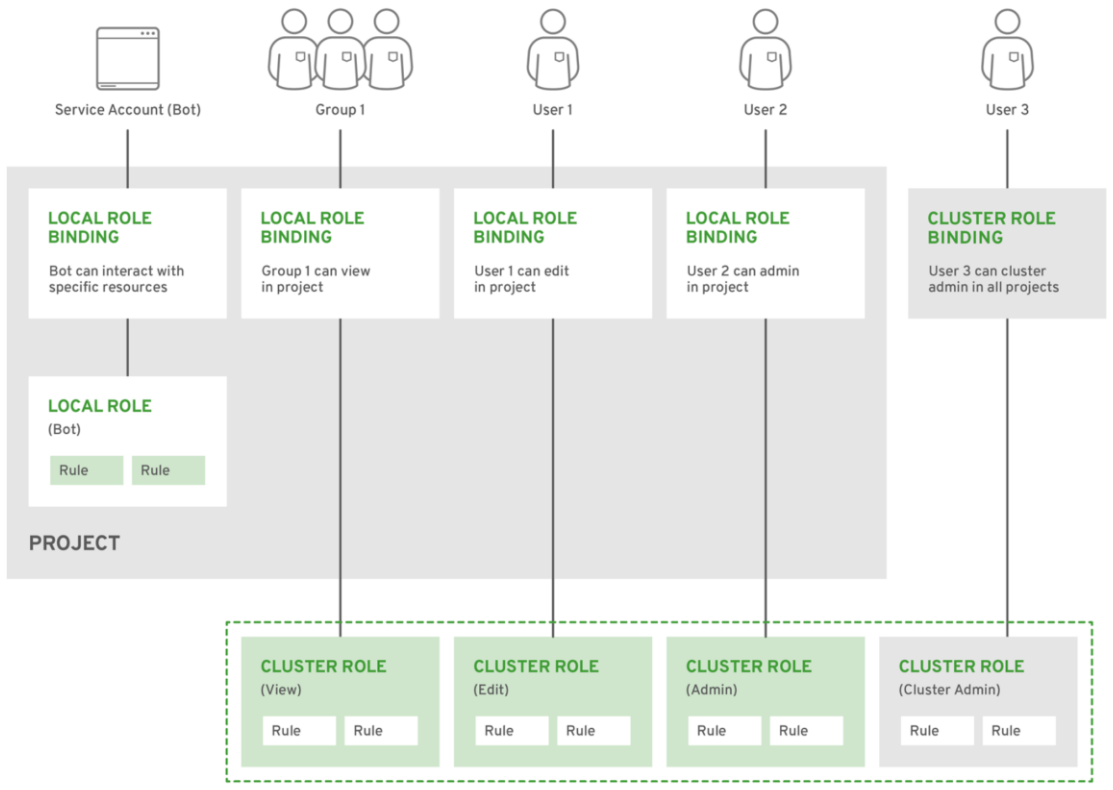


Figure 13 - Authorization Relationships

Several factors are combined to make the decision when OpenShift evaluates authorization: Identity, Action, and Bindings. The following steps are taken when OpenShift evaluates authorizations:

1. The identity and project-scoped action are used to find all user and group bindings.
2. Bindings are used to locate all the roles that apply.
3. Roles are used to find all the rules that apply.
4. The action is checked against each rule to find a match.
5. If no matching rule is found, the action is then denied by default.

There are two levels of RBAC roles and bindings that control authorization: Cluster RBAC and Local RBAC. Cluster RBAC are the roles and bindings that are applicable across all projects. Roles that exist cluster-wide are considered cluster roles. Cluster role bindings can only reference cluster roles. Local RBAC are roles and bindings that are scoped to a given project. Roles that exist only in a project are considered local roles. Local role bindings can reference both cluster and local roles.

This allows for a dual-level hierarchy for evaluating authorization and allows for re-usability over multiple projects through the cluster roles while allowing customization inside of individual projects through local roles. The evaluation first evaluates cluster-wide “allow” rules, then local-bound “allow” rules, followed up by the default deny-all rule.

(**7.2.3**) The selection of identity provider for authentication is configured post-installation and defaults to deny all behavior if not specified.

7.3 Ensure that security policies and operational procedures for restricting access to cardholder data are documented, in use, and known to all affected parties.

It will be the responsibility of the payment entity to document security policies and operational procedures for restricting access to CHD. This includes specific policies and procedures as it pertains to the access of the OpenShift environment, where it is used in support of CHD. The payment entity will be responsible for ensuring that the administrators and users of the OpenShift environment are aware of and adhere to their policies and procedures.

REQUIREMENT 8

Identify and Authenticate access to system components

OpenShift should be configured to work with a third-party identity provider, external to OpenShift, to address requirement 8 controls. Through the payment entity's chosen identity and authentication provider, unique identifiers can be setup for each user (**8.1.1**) prior to allowing the user to access components of the OpenShift environment. OpenShift can integrate with third-party identity providers through several mechanisms including LDAP. Control of identifier objects (**8.1.2**) should be performed with the chosen identity provider. Once the identity and authenticator(s) have been verified, the OAuth server built into the OpenShift Control Plane issues an OAuth access token to the user to allow for authentication to the API. When a person requests a new OAuth token, the OAuth server uses the configured identity provider to determine the identity of the person making the request. The OAuth server determines what user that the identity maps to, creates an access token for that user, and returns the token for use.

Revocation of access for terminated users would be performed with the identity provider. Users with revoked access would not be able to access OpenShift. (**8.1.3**) Likewise, removal or disabling of inactive user accounts within 90 days (**8.1.4**) would be handled with the identity provider. All user IDs, including those handled by third parties to access, support, or maintain system components via remote access, would be handled externally to OpenShift. The options for controlling remote access for third parties is the responsibility of the payment entity (**8.1.5**).

Account lockout for failed attempts would be managed by the identity provider as all authentication attempts that occur prior to granting access from OpenShift. Establishing a threshold for limiting repeated failed attempts would be configured with the chosen identity provider (**8.1.6**). Likewise, the lockout duration for the account and mechanisms to unlock the account for use would be established with the identity provider (**8.1.7**).

Session timeouts (**8.1.8**) can be enabled with OpenShift to limit the amount of time that a session can be active. It is, however, recommended that the payment entity control idle session timeouts at the user or administrator endpoint, rather than at the OpenShift console.

The type of authenticators to be used (for example, password or passphrase, token device or smart card, or biometrics) are also managed externally to OpenShift by the identity provider (**8.2**). The protection of the authentication credentials such as rendering the passwords and passphrases unreadable during transmission and the storage of credentials on system components is the responsibility of the third-party identity provider (**8.2.1**). Likewise, modification of authentication credentials is handled by the third-party identity provider (**8.2.2**). All access to modify parameters for authentication tokens or for generating keys within OpenShift is managed with RBAC and requires prior authentication before the user is authorized to act.

Parameters for authenticators such as password length, maximum password age, minimum password age, password history, and requirements to change the password on first use are also handled by the third-party identity provider (**8.2.3** – **8.2.6**).

Where multi-factor authentication is required, this also occurs outside of OpenShift (**8.3**).

Payment entities are required to communicate policies and procedures pertaining to identity and authentication (**8.4**). The payment entity is also required to not use group, shared, or generic IDs, passwords, or other authentication methods (**8.5**). Access tokens that are issued by OpenShift upon authentication should only be used by the person for whom it was issued.

Access tokens can be configured to have an expiration and require re-authentication with the identity provider to issue a new access token. The OAuth server generates two kinds of tokens: access tokens and authorize tokens. Access tokens are longer-lived tokens that grant access to the API. Authorize tokens are short-lived tokens whose only use is to be exchanged for access tokens. Token options can be set to establish an age limit for each type of token.

When using LDAP authentication for an OpenShift Container Platform cluster, one must create a custom resource (CR) for the identity provider and connect to a previously configured ConfigMap object in the openshift-config namespace, which contains the certificate authority bundle. Using a previously defined LDAP secret, the cluster can be configured to validate user names and passwords against an LDAPv3 server, using simple bind authentication. If failover is required for the LDAP server, use the steps provided by Red Hat documentation to extend the basis authentication method by configuring SSSD for LDAP failover. During authentication, the LDAP directory is searched for an entry that matches the provided user name. If a single unique match is found, a simple bind is attempted using the distinguished name (DN) of the entry plus the provided password.

The LDAP configuration can be configured to use TLS for connections to the server. Administrators can create a whitelist for users for an LDAP integration using the lookup mapping method. Before a login from LDAP would be allowed, a cluster administrator would need to create an identity and user object for each LDAP user.

REQUIREMENT 9

Restrict physical access to cardholder data

The payment entity is responsible for the secure placement of the OpenShift environment within the confines of a physical location whereby the payment entity can enable controls to restrict physical access to OpenShift, its components, and the supported workloads.

REQUIREMENT 10

Track and monitor all access to network resources and cardholder data

OpenShift has the means to generate audit logs that can be used to track access and actions taken by users and services within OpenShift. These logs can be sent to or consumed by a security information and event monitoring (SIEM) solution external to OpenShift. Much of the requirements of requirement 10 will be controlled external to OpenShift.

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

All actions taken by users of OpenShift are logged and capable of being used to satisfy audit requirements.

10.2 Implement automated audit trails for all system components to reconstruct the following events:

10.2.1 All individual users accesses to cardholder data

All user and/or service account accesses to OpenShift components are logged. The payment entity would be responsible for enabling logging for access to applications within workloads hosted in containers in OpenShift.

10.2.2 All actions taken by any individual with root or administrative privileges

All actions taken by individual with root or administrative privileges to OpenShift and Red Hat CoreOS are logged.

10.2.3 Access to all audit trails

Access to audit trails relative to OpenShift are made available at the OS level with administrator accounts. Red Hat CoreOS can be configured to log access to the journal or log file. For better protection of audit trails, including improved access controls, it is recommended to direct logs to an external log server or Security Information Event Management (SIEM) solution.

10.2.4 Invalid logical access attempts

Invalid logical access attempts pertaining to incorrect input of credentials would be handled by the payment entity's chosen identity provider. Unauthorized attempts to access system components or run unauthorized commands against OpenShift are logged.

10.2.5 Use of and changes to identification and authentication mechanisms – including but not limited to creation of new accounts and elevation of privileges – and all changes, additions, or deletions to accounts with root or administrative privileges.

Like 10.2.4, changes to identification and authentication mechanisms would be handled by the payment entity's chosen identity provider. Changes that are made to RBAC within OpenShift are logged. These logged events may be an indication of attempts to modify defined roles to grant additional privileges.

10.2.6 Initialization, stopping, or pausing of audit logs.

Stopping the mechanisms for log creation in OpenShift requires stopping the OpenShift Control Plane itself, which would have the effect of preventing any further access for any users to the API, CLI, or Web UI. Auditing within OpenShift cannot be reconfigured or stopped without reconfiguring OpenShift. Any attempt to reconfigure OpenShift will be logged.

10.2.7 Creation and deletion of system-level objects

Creation and deletion of system levels objects is logged by OpenShift (for OpenShift objects) and by Red Hat CoreOS.

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

The logs generated by OpenShift and Red Hat CoreOS include user identification (**10.3.1**), type of event (**10.3.2**), date and time of the event (**10.3.3**), success or failure indication (**10.3.4**), origination of event (**10.3.5**), and the identity or name of affected data, system component, or resource (**10.3.6**).

10.4 Use time-synchronization technology, synchronize all critical system clocks and times and ensure that the following is implemented for acquiring, distributing, and storing time. Note: One example of time synchronization technology is Network Time Protocol (NTP).

OpenShift uses time from the host OS. Red Hat CoreOS can be setup as an NTP client to receive time from the payment entity's chosen NTP server. Time synchronization should be setup on all hosts in the cluster whether using NTP or another method. This allows the systems to have the correct and consistent time, which is necessary for the proper functioning of OpenShift (**10.4.1**). Time data is protected (**10.4.2**) as it is part of the underlying OS that is obfuscated from the OpenShift user interfaces.

It is recommended that the payment entity use industry accepted time sources as the source for time synchronization (**10.4.3**). Typically, payment entities will opt to use a local time (NTP) server for synchronization of internal resources. The local time server would be configured by the payment entity to synchronize with a trusted external source using the NTP protocol.

10.5 Secure audit trails so that they cannot be altered.

It is recommended to use an external log aggregation solution or SIEM solution for securing audit trails. While the logs reside on the Red Hat CoreOS server, access can be controlled using RBAC. An external solution may be better equipped to secure audit trails in alignment with compliance requirements. RBAC controls in Red Hat CoreOS can be used to limit the ability to review audit logs and journals. An external solution may be able to provide improved granularity as well as search capability that would be of better use to the payment entity to satisfy requirements (**10.5.1**). Limited access to the audit trails on OpenShift hosts provides minimal protection from unauthorized modification. Use of an external log collector or SIEM solution may provide improved protections against unauthorized modifications by adding additional features such as file integrity monitoring, digital signing, or Write Once, Read Many (WORM) storage (**10.5.2 – 10.5.5**). Likewise, an external resource may be better equipped to manage and control retention (**10.5.7**).

REQUIREMENT 11

Regularly test security systems and processes

The payment entity will be responsible for testing their security systems and processes. This should include security systems, constraint, and controls both configured within OpenShift and those systems that support or integrate with OpenShift. The payment entity should include the OpenShift environment in their scheduled vulnerability scans to identify any known vulnerabilities and address them in a timely manner. The payment entity should be routinely scanning images and image repositories for vulnerabilities.

The payment entity should include the OpenShift environment as well as the workloads running in OpenShift as targets for penetration testing to verify that security controls which prevent unauthorized access or system modifications are working sufficiently and cannot be bypassed either through known vulnerability or through back channels resulting from misconfigured or poorly configured hosts and/or containers.

Penetration testing should include segmentation testing to verify the sufficiency of segmentation controls enabled in OpenShift to control container network access and to isolate CDE systems containers from out-of-scope systems and to minimize and monitor access of supporting connected-to and security-impacting systems.

REQUIREMENT 12

Maintain a policy that addresses information security for all personnel

When integrating or adding new systems, payment entities should evaluate the policies and procedures to ensure that coverage is sufficient to address the nuances of the system being implemented. It is also important for personnel to understand the policies and procedures with respect to the new technology. This helps to increase awareness of potential risk, validate proper implementation of technology according to required compliance standards, and facilitate assignment and accountability for new roles and responsibilities.

CONCLUSION AND COALFIRE OPINION

Red Hat OpenShift 4.5 hosted on Red Hat CoreOS, as reviewed by Coalfire, **can be effective** in providing support for the outlined objectives and requirements of PCI DSS v3.2.1. Through proper implementation and integration into the organization's overall technical infrastructure and information management systems, OpenShift may be useable in a PCI DSS v3.2.1 controlled environment. Care should be given for the implementation of OpenShift and the use of the platform for the deployment of containers in support of micro-services architectures. The organization planning to use OpenShift should also consider available guidance from PCI SSC on cloud computing, which includes specific guidance on securing containerized workloads.

The OpenShift SDN provides the means to implement network segmentation for the isolation of workloads in a single OpenShift container cluster and to control the communication between containers within the cluster according to design. OpenShift Service Mesh can be deployed for additional security.

Coalfire's conclusion is based on observations and analysis of the provided documentation, interviews with Red Hat personnel, and hands-on engagement with and testing of a lab environment designed and architected for hosting CDEs. The provided conclusions are based upon several underlying presumptions and caveats, including adherence to vendor best practices and hardening of configuration as supported by the system components. This solution should be implemented in alignment with the organization's mission, values, business objectives, general approach to security and security planning, and with respect to the overall organizational security and compliance program.

A COMMENT REGARDING REGULATORY COMPLIANCE

Coalfire disclaims generic suitability of any product to cause a payment entity to use that product to achieve regulatory compliance. PCI entities attain compliance through a Governance, Risk Management, and Compliance (GRC) program, not via the use of a specific product. This is true for payment card entities subject to PCI DSS, as well as for customers targeting compliance with other regulations.

LEGAL DISCLAIMER

Coalfire expressly disclaims all liability with respect to actions taken or not taken based on the contents of this white paper and the supporting controls workbook and the opinions contained therein. The opinions and findings within this evaluation are solely those of Coalfire and do not represent any assessment findings, or opinions, from any other parties. The contents of this document are subject to change at any time based on revisions to the applicable regulations and standards (e.g., Health Information Portability and Accountability Act [HIPAA], PCI-DSS, et al.). Consequently, any forward-looking statements are not predictions and are subject to change without notice. While Coalfire has endeavored to ensure that the information contained in this document has been obtained from reliable sources, there may be regulatory, compliance, or other reasons that prevent Coalfire from doing so. Consequently, Coalfire is not responsible for any errors or omissions, or for the results obtained from the use of this information. Coalfire reserves the right to revise any or all of this document to reflect an accurate representation of the content relative to the current technology landscape. To maintain the contextual accuracy of this document, all references to this document must explicitly reference the entirety of the document inclusive of the title and publication date. Neither party will publish a press release referring to the other party or excerpting highlights from the document without prior written approval of the other party. For questions regarding any legal or compliance matters referenced herein, you should consult legal counsel, your security advisor, and/or the relevant standard authority.

ADDITIONAL INFORMATION, RESOURCES, AND REFERENCES

This section contains a description of the links, standards, guidelines, and reports used for the materials used to identify and discuss the features, enhancements, and security capabilities of OpenShift 4.5.

RED HAT

The Red Hat OpenShift Container Platform documentation used to provide depth and context for this document is available at the Welcome page. The left column of the page provides further details into every capability for OpenShift. These details are available at the following link: https://docs.openshift.com/container-platform/4.5/welcome/index.html.

PAYMENT CARD INDUSTRY SECURITY STANDARDS COUNCIL

A number of key documents are fundamental in their support for Payment Card Industry guidance and supplemental guidance and are used by QSAs and payment card entity security team alike. Primary guidance for SAQ and ROC assessment may be found in the *Payment Card Industry (PCI) Data Security Standard, Requirements and Security Assessment Procedures, Version 3.2.1, May 2018*, located here: https://www.pcisecuritystandards.org/document_library

Additional information in the form of information supplements with titles: *Information Supplement: Guidance for PCI DSS Scoping and Network Segmentation, May 2017*, and *Information Supplement: PCI DSS Virtualization Guidelines, June 2011*, is at: https://www.pcisecuritystandards.org/document_library and at https://www.pcisecuritystandards.org/documents/Guidance-PCI-DSS-Scoping-and-Segmentation_v1_1.pdf

The PCI SSC conducted a revision to their virtualization and containerization standards during a 2017-2018 Cloud Special Interest Group (Cloud SIG) development session which culminated in the publication of the *Information Supplement: PCI SSC Cloud Computing Guidelines, April 2018* which may be located here: https://www.pcisecuritystandards.org/pdfs/PCI_SSC_Cloud_Guidelines_v3.pdf

ABOUT THE AUTHORS AND CONTRIBUTORS

Chris Krueger | **Revision Author** | Principal II, Solutions Engineering, Coalfire Systems

As Principal, Mr. Krueger contributes as an author and thought leader on information security and regulatory compliance topics for Coalfire's clientele in new and emerging technical areas.

Jason Macallister | **Original Author** | Senior Consultant, Coalfire Systems, Inc.

Mr. Macallister consults on Information Security and regulatory compliance topics as they relate to advanced infrastructure, emerging technology, and cloud solutions.

Fred King | **PCI SME** | Principal I, Payments, Coalfire Systems, Inc.

Mr. King is responsible for client service delivery with an emphasis on Security Architecture. Fred brings three decades of technical and management experience. Fred has designed and delivered IT security, networking, and virtualization architectures in both internal and professional services roles. Fred is a leader, a trained and experienced technologist, a security architect, and a QSA in the payment card industry.

Published Q1, 2021.

ABOUT COALFIRE