How Styra Maps to PCI Data Security Standard v3.2

Introduction and PCI DSS Overview

Created to facilitate consistent data security measures for cardholder data, the Payment Card Industry Data Security Standard (PCI DSS) works to provide a baseline of technical and operational requirements designed to protect data. PCI DSS applies to all entities involved in payment card processing—including merchants, processors, acquirers, issuers, service providers, and other entities that store, process or transmit cardholder and sensitive data. The 12 high-level requirements are broken down into functional testing procedures which help organizations implement the proper policies for handling sensitive data. Below are the 12 high-level requirements.

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.

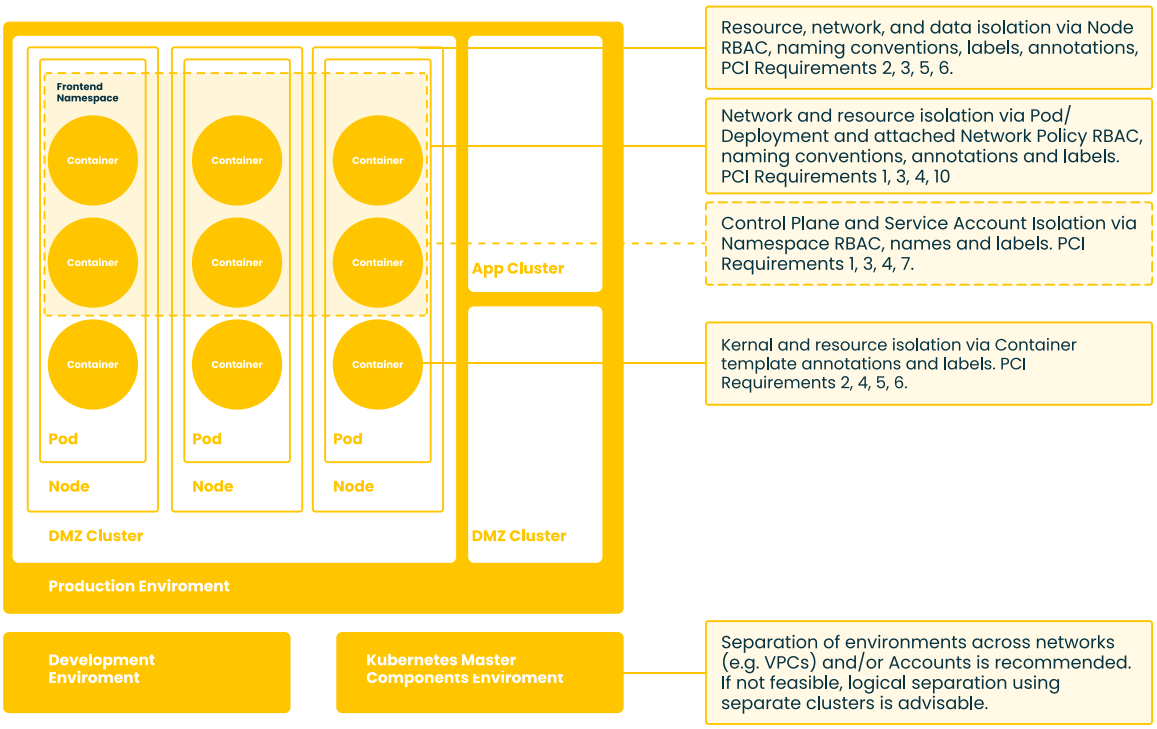
Cloud-native has transformed the way organizations do business and redefined network and storage parameters. From the infrastructure that maintains IT operations to the applications that supply customers with the ability to interact with their data - the velocity in which DevOps teams have to deliver these services has significantly increased, leaving little to no room for error. Styra enables organizations to maintain their desired velocity by ensuring that teams can quickly and efficiently validate compliance, maintain clear logs for audit, test policies before production to reduce down-time, and provide a unified control plane for cross-team collaboration.

We will highlight how each Requirement maps to Kubernetes, the best way to address them in PCI DSS v3.2, and where Styra has pre-built policies. Styra, like most vendors does not cover all 12 of the high-level PCI DSS requirements, but we do map to 1, 2, 3, 4, 5, 6, 7, and 10 - with 40 sub-requirements, supported by more than 150 pre-built policies to help organizations establish and maintain compliance. For the requirements that we do not map to, they are still critical to organizations and can be found in other solutions that span different areas of focus.

Kubernetes & PCI DSS Compliance

In the recent findings from Enterprise Strategy Group (ESG) comprising 500 IT and security personnel in North America and Western Europe in January 2021, the previous years events accelerated a massive move to the cloud with cloud-first organizations now outnumbering on-premise organizations by a ratio of three-to-one. Across these companies, there was a 200% jump in organizations planning to move more than 75% of their apps/workloads to the cloud, with 86% of companies placing cloud options in their decision process for new applications, and more than 40% choosing the cloud as their first option.

As more organizations begin and continue their cloud-native digital transformation, the importance of policy-as-code only increases. The quick rise has created a great need for regulatory compliance standards to adopt proper specifications for Docker/containers, orchestration/Kubernetes, or the kernel.



How Styra Can Help

Styra's Declarative Authorization Service (DAS) is purpose built by the founders of Open Policy Agent (OPA) to serve as a unified control plane throughout the policy lifecycle, which enables organizations to define, enforce and validate security across their cloud-native environments. Styra DAS provides a library of policies for PCI DSS compliance; these policies are enforced via a Kubernetes Admission Controller webhook interface.

Styra DAS then enables organizations to set up policies for Kubernetes and when Kubernetes receives an incoming resource for deployment, Styra DAS makes an authorization decision based upon a policy defined in a language called Rego. If the incoming Kubernetes resource (e.g. Node) is within the policy constraints, Styra DAS approves, otherwise, Styra DAS rejects or modifies it; therefore only compliance objects continue to go into production. Styra DAS works the way Kubernetes works - based on the Desired State principle, but applied to security.

Benefits

The sections below describe how Styra DAS policies map to specific PCI DSS control requirements for cloud-native and hybrid cloud compliance. For each of these requirements, Styra DAS has more than 150 pre-built policies mapping to 40 sub-requirements.

Pre-Mapped PCI DSS v3.2 Controls toKubernetes Policies

Manually mapping regulatory compliance standards to effective Kubernetes policies is a tedious task and often takes the expertise of a Qualified Security Assessor (QSA) that is qualified by the PCI Security Standards Council. Rather than spend invaluable resources and budget on assessing your policies in a static fashion, you can leverage Styra DAS to dynamically run your policies through a mapped library of PCI DSS rules.

Time Reduction on Audit Reporting

When the auditor shows up, that is when all productivity is halted to allocate resources to assist the auditor in the collection, understanding, and assessment of the policies that impact credit card data. With Styra DAS the amount of resources needed to assist the auditor is drastically reduced with automated compliance reporting for running clusters and a supplemental ongoing audit log of compliance decisions to prove compliance, without having to step through policy-as-code line-by-line.

Ensuring Comprehensive Controls

When there are a plethora of team members contributing to an application or applications, the need to maintain continuity of the desired outcome and establish true comprehensive controls can be daunting. With Styra DAS, organizations have access to out-of-the-box, easy to configure, policies that have been proven in the largest Kubernetes deployments on the planet, ensuring comprehensive control.

Cross-Team Collaboration

Kubernetes has introduced the need for multiple teams to be apart of the build process and with that comes non-coders and coders that need to be able to digest policies and their outcomes. Styra DAS, extends the ability to collaborate across multiple teams like DevOps, SOC, I&O (IT), and GRC with a single way for non-coders and engineers to collaborate and communicate about the state of security and compliance in Kubernetes.

Build and Maintain a Secure Network and Systems

Requirement 1: Install and maintain a firewall configuration to protect cardholder data.

While many organizations are still transitioning to a cloud-native environment and leveraging firewalls as a means to control north-south traffic, in time, firewalls will become obsolete. Where applications and data used to be solely reliant on data centers as a delivery mechanism for corporate networks, it is simply just not the scenario for most today. Today, applications are in cloud and hybrid environments and delivered in various capacities making them more accessible to the end user. Where this becomes a problem is when an organization relies on their firewall for traffic indicators.

Outlined below are ways to harden your configurations based on pre-built Policy Packs that map to sub-requirements of PCI DSS, allowing Styra DAS users the ability to isolate traffic, use ingress and egress selectors for traffic restriction, and much more that impacts how sensitive card holder data is processed, held, and maintained.

PCI DSS Requirements	Styra Policy Pack Action
1.1.1 A formal process for approving and testing all network connections and changes to the firewall and router configurations	Network Policy changes should only be allowed by a whitelist of authorized users.
1.1.4 Requirements for a firewall at each Internet connection and between any demilitarized zone (DMZ) and the internal network zone.	Use namespaces and labels in NetworkPolicy selectors to separate and isolate traffic between the public facing internet, DMZ web and API pods, internal applications, and highly sensitive card holder databases.
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.	Use naming conventions, annotations and labels on Pods and in corresponding NetworkPolicy selectors to explicitly identify and control network communication for specific business purposes.
1.1.7 Requirement to review firewall and router rule sets at least every six months.	Use annotations in NetworkPolicy selectors to explicitly record review status of each NetworkPolicy resource.
1.2 Build firewall and router configurations that restrict connections between untrusted networks and any system components in the cardholder data environment.	Use IP whitelists and blacklists to control network placement. Control ingress resources.
1.2.1 Restrict inbound and outbound traffic to that which is necessary for the cardholder data environment, and specifically deny all other traffic.	Control ingress resources and use namespaces together with ingress and egress selectors in NetworkPolicies to restrict traffic.
1.3.1 Implement a DMZ to limit inbound traffic to only system components that provide authorized publicly accessible services, protocols, and ports.	Use Service namespaces and labels to apply specific DMZ NetworkPolicies.

1.3.2 Limit inbound Internet traffic to IP addresses within the DMZ.	Use Service namespaces and labels to apply specific DMZ NetworkPolicies.
1.3.3 Implement anti-spoofing measures to detect and block forged source IP addresses from entering the network.	Restrict CAP_NET_ADMIN and CAP_NET_RAW capabilities in pods. NOTE: May be CNI specific.
1.3.4 Do not allow unauthorized outbound traffic from the cardholder data environment to the Internet.	Use egress selectors in NetworkPolicies to restrict traffic.
1.3.6 Place system components that store cardholder data (such as a database) in an internal network zone, segregated from the DMZ and other untrusted networks.	Use IP address whitelists and blacklists to control network placement. Use ingress whitelists.

Requirement 2: Do not use vendor-supplied defaults for system passwords and other security parameters.

In a recent report by Accurics, the Cloud Cyber Resilience Report, insecure defaults in Kubernetes made up nearly half of security violations due to insecure defaults — the most common type being improper use of the default namespace. For Requirement 2, outlined below is how the pre-built Strya DAS PCI Policy Packs can help organizations automate the process of ensuring that Kubernetes default passwords and settings are hardened, watching for lateral movement, and that a private key or certificate is present for specific actions.

PCI DSS Requirements	Styra Policy Pack Action
2.1 Always change vendor-supplied defaults and remove or disable unnecessary default accounts before installing a system on the network.	Implement CIS Benchmark and NIST 800-59 recommendations to harden Kubernetes.
2.2.1 Implement only one primary function per virtual system component.	Require labels for function with respect to system components that are in scope for PCI DSS, for example: Role/ClusterRole, RoleBinding/ClusterRoleBinding, Deployment, Service, Ingress, ServiceAccount, ConfigMap, Volume, PersistentVolume, etc.
2.2.2 Enable only necessary services, protocols, daemons, etc., as required for the function of the system.	Specify a trusted repository for all images, and specify specific image names corresponding to specific functions of the system, with the minimum necessary services, protocols, daemons, etc. Also restrict mounting paths that are sensitive.
2.2.3 Implement additional security features for any required services, protocols, or daemons that are considered to be insecure.	Secure all Ingress resources by specifying a Secret that contains a TLS private key and certificates.

2.2.4 Configure system security parameters to prevent misuse.	Require effective PodSecurityPolicies to prevent misuse and lateral movement of attackers.
2.2.5 Remove all unnecessary functionality, such as scripts, drivers, features, subsystems, file systems, and unnecessary web servers.	Specify trusted repositories and hardened images. Prevent pulling the "latest" from the repository since that can include newly added features that require thoughtful consideration and testing, undesired services and capabilities that have not been hardened. Only reviewed and tested images should be pulled.

Admission Control to the Rescue

Requirement 3: Protect stored cardholder data.

Encryption of cardholder data is more essential with the implementation of cloud-native stacks and Styra DAS provides policies ensure that a robust and compliant Key Management Server (KMS) is used in production by requiring consistent configuration and specific approved KMS endpoints in the EncryptionConfiguration resources.

PCI DSS Requirements	Styra Policy Pack Action
3.1 Keep cardholder data storage to a minimum by implementing data retention and disposal policies, procedures and processes.	Control how container state is handled after your containers exit. Enforce storage class configuration that supports correct delete functionality. Ensure all containers specify both CPU and memory requirements.
3.5 Implement procedures to protect keys used to secure stored cardholder data against disclosure and misuse.	Only allow key management service (KMS) providers with approved names and corresponding endpoints.
3.5.2 Restrict access to cryptographic keys to the fewest number of custodians necessary.	Prohibit unencrypted (identity) secret data storage.
3.6.1 Generation of strong cryptographic keys.	Use annotations to include the algorithms and properties so that strong encryption is enforced.
3.6.3 Secure cryptographic key storage.	Only allow key management service (KMS) providers with approved names and corresponding endpoints. Prohibit unencrypted (identity) secret data storage.
3.6.5 Key retirement or replacement.	Rotation of keys should be performed at defined frequencies. Currently this is a manual process in Kubernetes. Use Annotations on EncryptionConfigurations and Secrets to clearly identify when keys were last rotated and then monitor for rotation or replacement at specified intervals.

3.6.7 Prevention of unauthorized substitution of cryptographic keys.	Only allow encryption configurations to be created by approved users and groups.

Requirement 4: Encrypt transmission of cardholder data across open, public networks.

Cloud-native environments, especially Kubernetes are rife with cardholder data-at-rest and in-motion and the transmission of this over unencrypted networks. Styra DAS PCI DSS Policy Packs have a specific policy to ensure all ingresses have TLS configured.

PCI DSS Requirements	Styra Policy Pack Action
4.1.1 Ensure networks transmitting cardholder data or connected to the cardholder data environment, use industry best practices to implement strong encryption for authentication and transmission.	Ensure all ingresses have TLS configured.

Maintain a Vulnerability Management Program

Requirement 5: Protect all systems against malware and regularly update anti-virus software or programs.

A large majority of organizations use anti-virus software and with PCI DSS it is required, it is also recommended that the solution include anti-malware to supplement the protection of systems. Styra DAS Policy Packs enable the ability to write all malware and anti-virus configuration and signature files to read only filesystems and only allow images that have been appropriately scanned for viruses or malware.

PCI DSS Requirements	Styra Policy Pack Action
5.1.1 Ensure that anti-virus programs are capable of detecting, removing, and protecting against all known types of malicious software.	Require the use of trusted and secure repositories and only allow images that have been appropriately scanned for viruses or malware. Also do not pull the latest unscanned image.
5.2 Ensure that all anti-virus mechanisms are maintained.	Require the use of trusted and secure repositories and only allow images that have been appropriately scanned for viruses or malware. Also do not pull the latest unscanned image. Write all malware and antivirus configuration and signature files to read only filesystems.
5.3 Ensure that anti-virus mechanisms are actively running and cannot be disabled or altered by users.	Write all malware and antivirus configuration and signature files to read only filesystems.

Requirement 6: Develop and maintain secure systems and applications.

Organizations often push to production before they have evaluated their code against regulatory compliance frameworks. Implementing a policy-as-code process, can reduce the risk associated with runtime exploits that can compromise cardholder data and potentially more. Outlined below is how the Styra DAS PCI DSS Policy Packs will help enforce separation of development and test nodes from production nodes, enforce RBAC, prevent test volumes from being mounted in production workloads, and more.

PCI DSS Requirements	Styra Policy Pack Action
6.2 Ensure that all system components and software are protected from known vulnerabilities by installing applicable vendor supplied security patches.	Require the use of trusted and secure repositories and only allow images that have been appropriately scanned for vulnerabilities and have the lates security patches. Also do not pull the latest unscanned image.
6.3.1 Remove development, test and/or custom application accounts, user IDs, and passwords before applications become active or are released to customers.	Require the use of trusted and secure repositories and only allow images that have been appropriately configured without shared or test user IDs or passwords. Restric ServiceAccounts in reserved Namespaces. Restrict ClusterRoles with wildcards. Restrict which users and roles can create ClusterRoles. Restrict the storage of usernames or passwords in ConfigMaps.
6.4.1 Separate development/test environments from production environments, and enforce the separation with access controls.	Use node selectors and tolerations to enforce separation of development and test nodes from production nodes.
6.4.2 Separation of duties between development/test and production environments.	Enforce labels that identify the function as development, test, or production for all workloads and resources, ie. Pods/StatefulSets/Deployments, Services, Ingresses, Volumes, ConfigMaps, and Secrets. Use these labels for RBAC enforcement.
6.4.3 Production data (live PANs) are not used for testing or development.	Enforce labels that identify the function as test or production for all data resources, ie. require labels on PersistentVolumes, PersistentVolumeClaims, Pod/Deployment/StatefulSet. Use these labels to prevent test volumes from being mounted in production workloads.
6.4.4 Removal of test data and accounts from system components before the system becomes active / goes into production	Enforce labels that identify the function as test or production for all data resources, ie. require labels on PersistentVolumes, PersistentVolumeClaims, Pod/Deployment/StatefulSet. Use these labels to prevent test volumes from being mounted in production workloads. Enforce labels that identify the function as development, test, or production for all workloads and resources, ie. Pods/StatefulSets/Deployments, Services, Ingresses, Volumes, ConfigMaps, and Secrets. Use these labels for RBAC enforcement.

Implement Strong Access Control Measures

Requirement 7: Restrict access to cardholder data by business need to know.

Styra DAS Policy Packs augment default Kubernetes RBAC roles by prohibiting roles from being abused and requiring a consistent and well defined structure that can be traced to detailed human job classifications and responsibilities; thus implementing a "least privileged" control plane.

PCI DSS Requirements	Styra Policy Pack Action
7.1.2 Restrict access to privileged user IDs to least privileges necessary to perform job responsibilities.	Require labels on Roles/ClusterRoles for each job responsibility. Minimize privileges assigned to roles (e.g. no wildcards). Restrict reserved or sensitive patterns/prefixes on Roles/ClusterRoles, e.g. "system:". Prohibit roles and cluster roles from being created with the capability to access pod shells.
7.1.3 Assign access based on individual personnel's job classification and function.	Require labels on RoleBindings/ClusterRoleBindings for each job responsibility. Deny a ClusterRoleBinding to special "bootstrap" users or other sensitive users, e.g. cluster-admin user.
7.2.2 Assignment of privileges to individuals based on job classification and function.	Require labels on RoleBindings/ClusterRoleBindings for each job responsibility. Deny a ClusterRoleBinding to special "bootstrap" users or other sensitive users, e.g. cluster-admin user.

Regularly Monitor and Test Networks

Requirement 10: Track and monitor all access to network resources and cardholder data.

The ability to understand where cardholder data resides, who is accessing it, and when it was accessed is vitally important to preventing, detecting, and reducing risk for exposure. Additionally, in the event that an organization is compromised, logs allow thorough analysis to properly triage an incident. Outlined below is how Styra DAS Policy Packs ensure that actions only allow images that have appropriately configured NTP clients.

PCI DSS Requirements	Styra Policy Pack Action
10.4 Using time-synchronization technology, synchronize all critical system clocks and times and ensure that the following is implemented for acquiring, distributing, and storing time.	Require the use of trusted and secure repositories and only allow images that have appropriately configured NTP clients.
10.5.1 Limit viewing of assessment trails to those with a job-related need.	Require the use of trusted and secure repositories and only allow images that have appropriately configured NTP clients.
10.5.2 Protect assessment trail files from unauthorized modifications.	Require the use of an off cluster audit sink sending data encrypted via TLS.

Cloud-native application infrastructure allows for codifying policy as code, and implementing rules at every level of the stack, which changes the way teams must implement well-proven industry standards and regulations like PCI DSS. However, even the process of mapping regulations to new and fast-evolving technologies like Kubernetes, ServiceMesh, and Cloud configuration takes time, as traditional security and governance teams have to work closely with DevOps to identify where and how to codify rules to be effective.

About Styra

We are reinventing policy and authorization for cloud-native. Today's cloud app infrastructure has evolved. Access, security, and compliance must also evolve. It's time for a new paradigm. It's time for authorization-as-code.

Learn more at www.styra.com

styra