Architecting Amazon EKS for PCI DSS Compliance

June 25, 2021

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

© 2021 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Contents

Abstract

Companies are increasingly adopting the use of microservices and containers within AWS to support their sensitive data workloads. This paper outlines best practices customers should consider when configuring Amazon Elastic Kubernetes Service (Amazon EKS) for AWS Fargate or Amazon Elastic Compute Cloud (Amazon EC2) launch types for Payment Card Industry Data Security Standard (PCI DSS) version 3.2.1 compliance. It is not comprehensive of all of the PCI DSS controls as some are not applicable to containers and because customer environments vary.

The intended audience is system architects, developers, security personnel, and risk and compliance personnel who are interested in architecting their AWS Cloud environments for PCI DSS compliance.

This document was developed by AWS Security Assurance Services, LLC (AWS SAS), which is a fully owned subsidiary of Amazon Web Services. AWS SAS is an independent PCI Qualified Security Assessor (QSA) company (QSAC) that provides AWS customers and partners with specific and prescriptive information for achieving PCI DSS compliance in the AWS Cloud.

Introduction

The Payment Card Industry Data Security Standard (PCI DSS) provides technical and operational guidance on securing payment card processing environments that is applicable to people, processes, and technology. Entities that store, process, or transmit cardholder data (CHD) must validate compliance of their cardholder data environment (CDE) against the PCI DSS. Examples of such entities include merchants, payment processors, and service providers.

AWS provides many services that have been attested to have met PCI DSS compliance. You can refer to AWS Artifact which is a central resource for compliance-related information that provides on-demand access to AWS' security and compliance reports and select online agreements. Companies can leverage these services to reduce compliance efforts. One area of continued growth is the use of AWS containerized solutions.

A service listed as PCI DSS compliant does not mean that by default the use of that service makes a customer's environment compliant. Rather, it means the service has the ability to be configured to meet PCI DSS requirements. Where parameters are accessible and configurable by customers, it is the customer's responsibility to ensure they are configured to meet compliance requirements. There may be additional AWS services that are not included in the AWS PCI DSS assessment that can still be used to meet PCI DSS controls.

AWS container solutions include our managed service, Amazon Elastic Container Service (Amazon ECS), and Amazon Elastic Kubernetes Service (Amazon EKS). Each service supports deployment on either AWS Fargate or Amazon Elastic Compute Cloud (Amazon EC2). AWS Fargate is a serverless compute engine and removes the need to provision and manage EC2 instances. For Amazon EC2 deployment, customers manage the underlying EC2 instances running the containers.

The benefits of transitioning workloads to container services include platform independence, deployment speed, and resource efficiency. But it's important, as with any cloud workloads, to understand how to architect for security in containers. The transient and dynamic nature of container environments creates a continuously changing CDE that may make it difficult to assess.

Attack vectors for containerized applications are similar to those faced by non-container-based application deployments with the addition of the container management layer. As with other application deployments, we recommend that you continue to

operate within best practices, including adherence to Open Web Application Security Project (OWASP) concerns.

Container functions are typically architected to perform primary tasks, which in turn creates a distributed environment. The services implemented by containers become more network interdependent and require scheduling, scaling, and resource management. Unlike virtual machines, containers share the operating system's kernel. This setup can provide a common point of attack that can be leveraged to access all containers for a given host. When running multiple containers on a single operating system, all of the containers may share a common network interface. In this whitepaper, we will discuss the various solutions that you can build around AWS services to mitigate this security risk.

PCI DSS compliance status of AWS services

AWS is a Level 1 PCI DSS Service Provider, which enables our customers to more easily meet compliance requirements. The scope of the PCI DSS assessment assumes that for each service, any data provided by the customer could include primary account numbers (PAN) and sensitive authentication data (SAD), or impact the security of such data. The assessment also includes all physical security requirements that apply to AWS datacenters that support PCI DSS in-scope services.

The AWS Services in Scope by Compliance Program website lists the AWS services that were included in the annual PCI DSS assessment, along with all other services by compliance program. AWS service compliance is continually maintained and new features are assessed to determine if they can inherit the compliance status of the parent. Customers can access AWS compliance documentation through the AWS Management Console using AWS Artifact.

AWS Shared Responsibility Model

Security and Compliance is a shared responsibility between AWS and the customer. The Shared Responsibility Model helps relieve the customer's operational burden as AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the service operates.

The following figure provides an overview of the Shared Responsibility Model. The line of responsibility may vary depending upon the implemented AWS service.

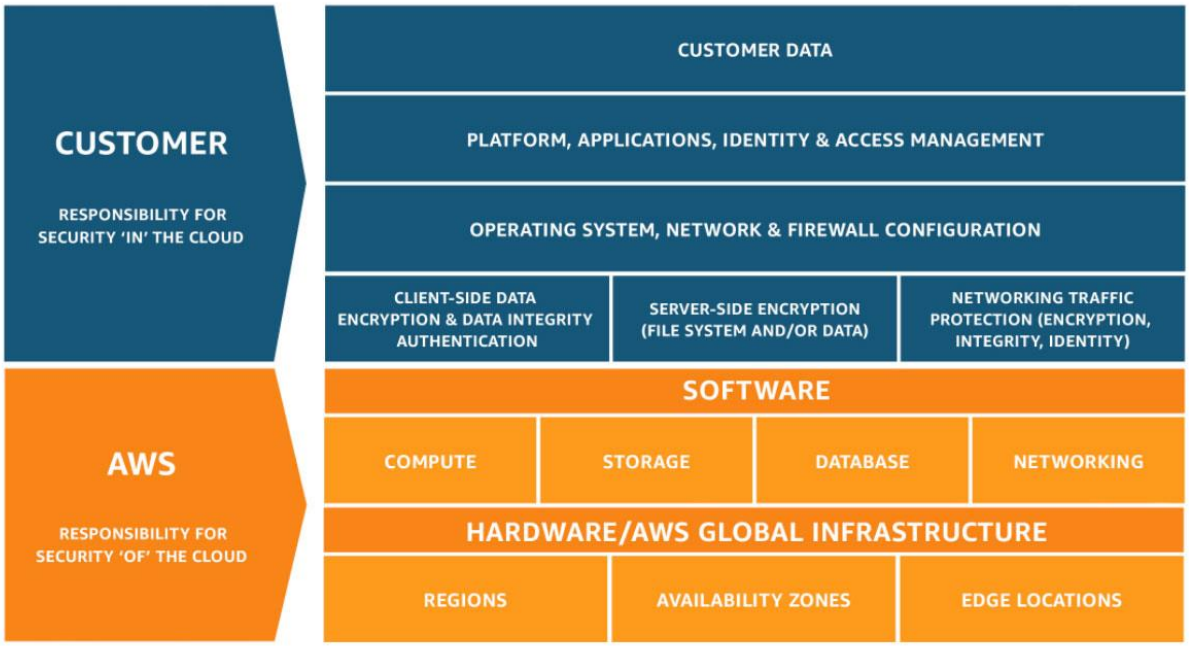

Figure 1 - Shared Responsibility Model

AWS is responsible for the security and compliance **of** the Cloud, which refers to the infrastructure that runs all of the services offered in the AWS Cloud. Cloud security at AWS is our highest priority. AWS customers benefit from a data center and network architecture that are built to meet the requirements of the most security-sensitive organizations. This infrastructure consists of the hardware, software, networking, and facilities that run AWS Cloud services.

Customers are responsible for the security and compliance **in** the Cloud, which consists of customer configured systems and services provisioned on AWS. For PCI DSS compliance, customer responsibility is all system components that includes AWS resources included in or connected to their CDE. For abstracted services, such as Amazon Simple Storage Service (Amazon S3) or Amazon DynamoDB, responsibility includes customer-configurable controls such as access controls, log settings, and encryption settings.

The division of responsibility will depend on the AWS service and implementation that the customer selects. Amazon EKS is a good example that allows customers to choose a serverless deployment of containers with AWS Fargate, or run containers on Amazon EC2 infrastructure that is accessible by the customer. With AWS Fargate, customers are abstracted from the underlying host and are not responsible for updating or patching the host system. Whereas an Amazon EKS deployment on Amazon EC2 requires

customers to take on a greater role of responsibility, such as controlling access and applying security patches.

If a customer can control a service parameter, they are responsible for ensuring it is configured to meet PCI DSS requirements.

PCI DSS scope determination and validation

It is critical is to understand the complete flow of cardholder data (CHD) within the environment. The CHD flow determines the applicability of the PCI DSS, defines the boundaries and components of a CDE, and therefore the scope of a PCI DSS assessment. Accurate determination of the PCI DSS scope is key to defining the security posture of the assessed workload and ultimately a successful assessment. Customers must have a procedure for scope determination that assures its completeness and detects changes or violations of the scope. Typically, the following steps comprise the PCI DSS scope identification:

1. **Identify the CHD flow**. Define the lifecycle of CHD including the path of consumption or entry of CHD in your environment, the subsequent processing and storing of the CHD, and eventually the secure destruction, devaluation, or exit of the CHD from your environment.
2. **Identify all in-scope resources in your environment**. Identify the various types of AWS resources involved in receiving, processing, storing and/or transmitting CHD making up the CHD flow.
3. **Categorize the system**. Categorize systems into abstracted and infrastructure services. The scope identification and segmentation of those resources are based on different types of connection, namely infrastructure service (OSI Layer 3–4) connection and abstracted services (OSI layer 7).
4. **Design segmentation boundaries**. Design segmentation boundaries to ensure that all other AWS resources are properly segmenting them to exclude them from the PCI DSS scope.

The ephemeral nature of containerized applications provides additional complexities when considering a dynamically changing scope. As a result, customers need to maintain an awareness of all container configuration parameters to ensure compliance requirements are addressed throughout all phases of a container lifecycle, as will be illustrated in the following sections.

Securing an Amazon EKS Deployment

The following sections provide guidance on key topics that you should consider when architecting a container-based environment for PCI DSS compliance. These sections comprise the following categories.

- Network segmentation
- Host and container image hardening
- Data protection
- Restricting user access
- Event logging
- Vulnerability scanning and penetration testing

Each section provides an overview of the requirements along with best practice recommendations to achieve compliance. The guidance is not all inclusive, as each customer's environment is unique. Collectively, the following recommendations provide a defense-in-depth approach to securing a container-based environment.

Network segmentation

Controls within requirement 1 of the PCI DSS call for installing and maintaining a firewall to protect cardholder data and require that systems be protected from unauthorized access. Firewall, in this context, means inbound and outbound access must be restricted to only approved ports and services. Although the use of network segmentation is not a PCI DSS requirement, its usage is a highly relevant tool to reduce the scope of a customer's environment.

An individual AWS account provides the highest level of segmentation boundary that can be achieved on the AWS platform. By design, all resources provisioned within an AWS account are logically isolated from resources provisioned in other AWS accounts, even within your own AWS Organizations. Use an isolated account for PCI DSS workloads to establish access segmentation when designing your PCI application to run on the AWS Cloud.

Amazon Virtual Private Cloud (Amazon VPC) and subnets will provide further logical isolation of CDE-related resources. You can deploy a VPC and subnets that meet the Amazon EKS requirements through manual configuration, or by deploying the VPC and subnets using eksctl, or an Amazon EKS provided AWS CloudFormation template.

Both eksctl and the AWS CloudFormation template create the VPC and subnets with the required configuration. For more information, see Creating a VPC for your Amazon EKS cluster.

By default, all pod-to-pod communication is allowed within a Kubernetes cluster. Kubernetes network policies provide a mechanism to restrict network traffic not only between pods but also between pods and external services. Kubernetes network policies operate at layers 3 and 4 of the Open Systems Interconnection (OSI) model. Network policies use pod selectors and labels to identify source and destination pods, but can also include IP addresses, port numbers, protocol numbers, or a combination of these. Calico is an open-source policy engine from Tigera that helps with network policy enforcement and works well with Amazon EKS. In addition to implementing the full set of Kubernetes network policy features, Calico supports extended network polices with a richer set of features, including support for layer 7 rules, e.g. HTTP, when integrated with service mesh like AWS AppMesh or Istio.

Calico policies can be scoped to namespaces, pods, service accounts, or globally. When policies are scoped to a service account, it associates a set of inbound/outbound rules with that service account. With the proper Kubernetes rule-based access control (RBAC) rules in place, you can prevent teams from overriding these rules, allowing IT security professionals to safely delegate administration of namespaces. When you first provision an EKS cluster, the Calico policy engine is not installed by default. The manifests for installing Calico can be found in the VPC CNI repository

Within AWS, security groups act as a virtual firewall and provide stateful inspection. You can use security groups to restrict communications by IP address, port, and protocol. It is important to note that, by default, security groups allow all outbound communications. As a result, you must configure outbound connection rules to meet PCI DSS requirements.

Amazon EKS uses Amazon VPC security groups to control the traffic between the Kubernetes control plane and the cluster's worker nodes. Security groups are also used to control the traffic between worker nodes, external IP addresses, and other VPC resources. It is strongly recommended that you use a dedicated security group for each control plane (one for each cluster). The minimum and suggested rules for the control plane and node group security groups can be found in Amazon EKS security group considerations.

To control communication between services that run within the cluster and services that run outside of the cluster, consider security groups for pods, which integrate Amazon EC2 security groups with Kubernetes pods. You can use Amazon EC2 security groups

to define rules that allow inbound and outbound network traffic to and from pods that you deploy to nodes running on many Amazon EC2 instance types. For a complete list of supported instances, see Amazon EC2 supported instances and branch network interfaces. Your nodes must be one of the supported instance types. Before deploying security groups for pods, consider the limits and conditions as discussed within the Amazon EKS User Guide, Security Groups for Pods.

AWS Fargate runs each pod in its own dedicated kernel runtime environment and does not share CPU, memory, storage, or network resources with other pods, which helps ensure improved workload isolation and security. However, because Kubernetes is a single-tenant orchestrator, potential pod-level intercommunication still exists. You should group sensitive workloads that need complete security isolation using separate Amazon EKS clusters.

> **Note:** At the time of writing, you cannot use security groups for pods running on Fargate. You must use Kubernetes network policy ingress/egress rules to control the traffic and achieve network segmentation.

Optimally, containerized workloads should be grouped based on data sensitivity levels to more easily facilitate network segmentation. Additionally, Kubernetes namespaces allow for resource segmentation inside the Kubernetes cluster with logical isolation from each other. Namespaces provide scope for pods, services, and deployments in the cluster, so that users interacting with one namespace will not see content in another namespace. However, namespaces within the same cluster don't restrict communication between namespaces. You need network policies for granular control and to restrict such communications between namespaces.

To summarize, consider the following when working to isolate containerized application communications:

- Isolate pods on separate nodes based on sensitivity of services and isolate CDE workloads in a separate cluster with a dedicated Security group.
- Use AWS security groups to limit communication between nodes and control plane and external communications.
- Implement micro-segmentation with Kubernetes network policies and consider usage of service mesh, Networking and Cryptography library (NaCl) encryption and Container Network Interfaces (CNIs) to limit and secure communications.

- Implement a network segmentation and tenant isolation network policy. Network policies are similar to AWS security groups in that you can create network ingress and egress rules. Instead of assigning instances to a security group, you assign network policies to pods using pod selectors and labels. For more information, see Installing Calico on Amazon EKS.

Host and image hardening

Requirement 2 of the PCI DSS emphasizes the need to disable support for vendor supplied defaults for system passwords and other security parameters. Amazon container services, like Amazon EKS, are run on container optimized Amazon Machine Images (AMI). These operating systems only contain additional libraries that are essential for container deployments, and as a result, help to minimize attack vectors.

Customers are still responsible for maintaining compliance of all configurations and functions at the operating system, network, and application layers. Operating systems should be routinely patched through the use of AWS Systems Manager whereas non-essential services and libraries should be disabled or removed. Configuration standards should be established that are consistent with industry-accepted system hardening guidelines, such as the Center for Internet Security (CIS) Benchmarks for EC2 instance types. Additional AWS secure configuration standards support is available on the AWS Security Learning website.

Consider using a special purpose operating system (OS) like Bottlerocket that includes a reduced attack surface, a disk image that is verified on boot, and enforced permission boundaries using SELinux.

Container builds should be limited to only required resources and adopt a model of microservices where a container provides one primary function. Software architects should ensure that images do not rely on outdated software libraries and applications that may contain known vulnerabilities. A best practice is to rebuild container images in the container registry on a periodic basis to ensure the latest application versions are in use. The usage of vulnerable libraries may introduce avenues of malicious activity that are often overlooked.

When managing containers, they should be immutable and not patched in-place. Customers should create trusted base container images that have been assessed and confirmed to use patched libraries and applications. Use a trusted registry to secure container images, such as Amazon Elastic Container Registry (Amazon ECR). Amazon ECR provides image scanning based upon the Common Vulnerabilities and Exposures (CVEs) database and can identify common software vulnerabilities.

Amazon Inspector can be leveraged as an automated security assessment service that helps improve the security and compliance of applications deployed on AWS. Amazon Inspector automatically assesses applications for exposure, vulnerabilities, and deviations from best practices.

Customers are responsible for ensuring that their Amazon EC2 instances run appropriate anti-virus and file integrity monitoring software when choosing the EC2 launch type for Amazon EKS. Many container vendors provide solutions optimized for container usage to address these requirements.

Alternatively, consider using the AWS Fargate launch type which provides on-demand, right-sized compute capacity for containers. With AWS Fargate, you no longer have to provision, configure, or scale groups of virtual machines to run containers. This option removes the need to choose server types, decide when to scale your node groups, or optimize cluster packing. Another advantage of using Fargate is that hardening, patching and monitoring the host system, and the worker node is taken care by AWS. Additional considerations for choosing Fargate can be found in the Amazon EKS User Guide.

We recommend that you consider implementing policy governance that can enforce security and compliance polices with tools like Gatekeeper, Open Policy Agent (OPA), and dockerfile-lint.

To summarize, consider the following points for host and image hardening:

- Use an OS optimized for running containers.
- Minimize access to worker nodes and deploy the worker nodes in private subnet.
- Run Amazon Inspector to assess hosts for exposure, vulnerabilities, and deviations.
- Use minimal container images and scan images for vulnerabilities regularly.

Data protection

The PCI DSS controls within requirements 3 and 4 are focused on the need to protect sensitive data while at rest and in transit. AWS provides a number of PCI DSS compliant services and features to assist with these compliance efforts.

Workloads that contain sensitive data, such as cardholder data, should secure all storage of data. Storage of data should be on secure file stores or databases and not on the underlying container host. System architects should be mindful of volume mounts

and sharing of data between containers, such as host file systems and temporary storage.

Make sure to secure sensitive data and environment variables, such as database connection strings that are contained within container build files. Many AWS services integrate with the AWS Key Management Service (AWS KMS), a PCI DSS compliant service that provides encryption key management functionality including secure encryption key storage, access controls, and annual rotation. AWS Secrets Manager and AWS Systems Manager Parameter Store are two services that can be used to secure sensitive data within container build files. AWS Systems Manager Parameter Store provides secure, hierarchical storage of data with no servers to manage. You can establish granular access and audit controls to help ensure appropriate restrictions are in place to meet compliance requirements. Data stored within AWS Systems Manager Parameter Store can be encrypted using AWS KMS.

Similar to AWS Systems Manager Parameter Store, data secured within AWS Secrets Manager also leverages the AWS KMS. AWS Secrets Manager provides additional capabilities that include random password generation and automatic password rotation. AWS KMS is a PCI DSS compliant service that is integrated with many AWS platform services. Users can create and manage cryptographic key material as well as control who can access and use the encryption keys.

For data in transit, PCI DSS requires that sensitive information be encrypted during transmission over open, public networks. Customers are responsible for configuring strong cryptography and security controls. AWS provides multiple services, such as Amazon API Gateway and Application Load Balancer, that support the use of Transport Layer Security (TLS). Policies can be applied to the services to enforce support of only TLS 1.1 or greater.

Amazon API Gateway and Application Load Balancer also support use of the integrated AWS WAF – Web Application Firewall to secure communications at the application-layer. AWS WAF helps protect applications and APIs against common web exploits like those identified within the OWASP Top 10.

Traffic exchanged between the Nitro instance types C5n, G4, I3en, M5dn, M5n, P3dn, R5dn, and R5n, is automatically encrypted by default except when there's an intermediate hop, such as an AWS Transit Gateway or a load balancer. In these instances, the traffic is not encrypted.

Encryption in transit for inter-pod communication can also be implemented with a service mesh like AWS App Mesh with support for mTLS.

Inbound traffic controllers are a way for you to intelligently route HTTP/S traffic that emanates from outside the cluster to services running inside the cluster. Often, inbound traffic is fronted by a layer 4 load balancer, such as the Classic Load Balancer or the Network Load Balancer. An inbound traffic controller can be configured to terminate SSL/TLS connections.

To summarize, consider the following points for data protection:

- Leverage AWS KMS for service-managed encryption keys and avail AWS managed CMKs or rotate your Customer Master Keys (CMKs) periodically
- Enable support for strong encryption in transit.
- Use envelope encryption of Kubernetes secrets in EKS to add a customer-managed layer of encryption for application secrets or user data that is stored within a Kubernetes cluster.
- User access

The controls within requirements 7 and 8 of the PCI DSS are focused on restricting access to authorized personnel and ensuring appropriate access controls are in-place. Access to resources should embrace a least privilege model where access is on a need-to-know basis. User access to containers and the underlying host should be authenticated with strong authentication requirements that align with the PCI DSS.

Container images should be run with non-privileged user accounts. For instance, container build files that do not contain defined user credentials will run as root by default. This setup means that a compromised container service may extend root privileges to an malicious actor who may use the elevated access to further exploit the underlying host.

Unlike AWS Fargate where no host access is available, Amazon EKS with EC2 launch type provide the option to enable secure shell (SSH) access for underlying system management. Consider disabling the use of secure shell (SSH) and instead leverage AWS Systems Manager's Run Command. With Run Command, there are no SSH keys to manage, and all invoked operations are auditable within AWS CloudTrail.

In an effort to create and establish secure container images, restrict all access to container images. Container deployments should use a private container registry that restricts access and write permissions, such as Amazon ECR, which integrates with AWS Identity and Access Management (IAM) for access controls. Amazon ECR is a scalable container repository that provides secure storage and transmission of container

images. The simplified workflow and integration of Amazon ECR with AWS services also reduces the need for excessively providing credentialed access to container hosts.

With Amazon EKS and its required IAM authenticator, users sign into the cluster with an IAM identity – either an IAM user or IAM role. Kubernetes then decides what actions the user can perform via its role-based access control (RBAC). You must configure AWS IAM roles to set up authorization at the cluster and infrastructure level. With RBAC, you can set up authorization to the resource level (e.g. particular pod) or service level (e.g. pod). Employ least privileged access when creating Roles/RoleBindings for namespace level resources and ClusterRole/ClusterRoleBindings for cluster level resources.

When you create an Amazon EKS cluster, the IAM entity user or role (such as a federated user that creates the cluster) is automatically granted system:masters permissions in the cluster's RBAC configuration. This access cannot be removed and is not managed through the aws-auth ConfigMap. Therefore, it is a best practice to create the cluster with a dedicated IAM role and regularly audit who can assume this role. This role should not be used to perform routine actions on the cluster, and instead additional users should be granted access to the cluster through the aws-auth ConfigMap for this purpose. For more information see Managing users or IAM roles for your cluster.

To summarize, consider the following points for user access:

- Employ least privileged access to AWS resources when creating RoleBindings and ClusterRoleBindings;
- Use IAM Roles when multiple users need identical access to the cluster and IAM Roles for Service Accounts (ISRA) where possible.
- Make the Amazon EKS Cluster endpoint private.
- Create the cluster with a dedicated IAM role which should be regularly audited.
- Regularly audit access to the cluster.
- Run the application as a non-root user.

Tracking and monitoring access

Event logging

The core control within requirement 10 of the PCI DSS is the need to use event logging mechanisms to track, monitor, and alert on potentially anomalous activities.

Leverage AWS event log services to establish event log monitoring at the network, host, and container level. Enable VPC Flow Logs to capture network traffic that details packet information, such as the protocol, port, and source and destination address information. Monitor container hosts to ensure health, efficiency, and availability by ensuring Amazon CloudWatch or Amazon Kinesis agents are enabled and configured.

Enable event logging capabilities within the containerized applications to capture application and container event log data. Use CloudWatch dashboard to monitor and alert on all captured event log activity. Store the captured event data securely within encrypted Amazon S3 buckets to help you meet your retention requirements.

Amazon EKS with Fargate supports a built-in log router, which means there are no sidecar containers to install or maintain. The log router allows you to use the breadth of services at AWS for log analytics and storage. You can stream logs from Fargate directly to Amazon CloudWatch, Amazon Elasticsearch Service, and Amazon Kinesis Data Firehose destinations such as Amazon S3, Amazon Kinesis Data Streams, and partner tools. Fargate uses a version of Fluent Bit, an upstream compliant distribution of Fluent Bit managed by AWS. For more information, see AWS for Fluent Bit on GitHub

Lastly, maintain a holistic view of the environment through the use of AWS tools. Amazon GuardDuty provides threat detection through anomaly detection, machine learning, and threat intelligence of events across AWS data sources, including AWS CloudTrail and VPC Flow Logs. Amazon Athena and Amazon CloudWatch Logs Insights can also be used to query and analyze audit trail logs saved to Amazon S3 from VPC Flow Logs, AWS CloudTrail, and Amazon CloudWatch.

To summarize, consider the following points for monitoring and logging:

- Enable EKS Cluster audit logs.
- Use Kubernetes audit metadata annotations for authorization history tracking.
- Create alarms for suspicious events.
- Analyze logs with Amazon CloudWatch Log Insights.
- Audit your AWS CloudTrail logs.

Network intrusion detection

Controls within requirement 11 of the PCI DSS specify the use of intrusion-detection and/or intrusion-prevention techniques to detect and/or prevent intrusions into the network. The standard requires monitoring of all traffic at the perimeter and critical

points of the CDE. With most on-premises environments, the requirements are typically met by using Intrusion Detection System (IDS)/Intrusion Prevention System (IPS) appliances. A similar approach can be used within AWS.

When considering containerized environments, inspection of network traffic can be done at the network layer outside of the container host and within the container management software's virtual container network.

There are several options that can be considered for inspection of network data outside of the container host on AWS. Amazon GuardDuty is a managed service that provides threat detection across multiple AWS data sources to identify threats. It uses machine learning, anomaly detection, and threat intelligence to help identify illicit network activity.

When considering a traditional IDS/IPS solution, Amazon VPC Traffic Mirroring can be configured to route a copy of all network communications to a virtual appliance running on one or more Amazon EC2 instances.

Another common solution is to use a transit network architecture that uses IP routing to ensure that all network traffic crosses a single network. This architecture allows you to use a virtual IDS/IPS device from the AWS Marketplace to inspect all traffic transiting between networks. It is possible to also use a VPC Gateway to route all traffic to on-premises IDS/IPS infrastructure. Lastly, host-based IDS or IPS solutions can also be used to inspect traffic as it is delivered to an Amazon EC2 instance.

Inspection of inter-container communications on the virtual container network is another viable option. There are vendors within the AWS Marketplace that provide IDS container solutions, which mostly use a side container to monitor and alert on unusual traffic patterns. Agent based solutions are also available that use machine learning to detect anomalous communication patterns among the containers.

The security measures put into place will depend heavily on the architecture of the environment. Traffic detection at the network layer will require advanced planning of container deployments and traffic patterns.

Vulnerability scanning and penetration testing

The PCI DSS control requirement 11.2 requires organizations to regularly test systems and processes to identify vulnerabilities and remediate such findings in a timely manner. Vulnerability scanning should be performed on a quarterly basis as well as after any significant changes to the environment. Similarly, penetration testing is to be performed on an annual basis and after any significant environment changes. Penetration testing of AWS resources is allowed at any time for certain permitted services. The AWS

support policy for penetration testing should be consulted for further details, or consult with your account team. For those service providers who are using network segmentation, they are required to test the effectiveness of segmentation controls every six months or after any changes to the segmentation controls.

The scope of the assessment activities will include the CDE and ancillary systems used in support of the CDE. See the PCI DSS Information Supplement: Penetration Testing Guidance for scope and methodology guidance when performing penetration testing.

Depending upon a customer's environment, the test requirements may apply to on-premises, cloud resources, and containerized environments. When deploying Amazon EKS on Amazon EC2 instances, customers must perform vulnerability scanning of the underlying host. Per the PCI DSS requirement, customers are responsible for establishing a process to identify security vulnerabilities, and assigning a risk ranking to newly discovered security vulnerabilities. Amazon Inspector is a security assessment tool that helps identify vulnerabilities and prioritizes findings by level of severity. Integration of Amazon Inspector within the DevOps process provides for assessment automation to proactively identify vulnerabilities and to check for unintended network accessibility of your nodes.

You can also use container specific scanning tools to scan container images for vulnerabilities. Container scanning identifies non-compliant code, vulnerable libraries, and potentially exposed secrets. Security vendors within the AWS Marketplace provide solutions capable of scanning systems, containers, and applications.

When performing internal and external penetration testing, assessment activities should be done at both a network and application layer and should target the underlying host and containerized applications. Patch container hosts to address vulnerabilities and update container images to mitigate identified container vulnerabilities. Create golden images for containers and securely store them within private container registries, such as Amazon ECR.

The Center for Internet Security (CIS) Kubernetes Benchmark provides guidance for Amazon EKS node security configurations. It can be run using kube-bench, a standard open source tool for checking configuration using the CIS benchmark on Kubernetes clusters. To learn more, see Introducing the CIS Amazon EKS Benchmark.

Conclusion

AWS provides multiple services to support customers' containerized workloads, and customers can configure the services to best meet their data processing needs. Because of this flexibility, organizations must maintain an awareness of all compliance requirements throughout the lifecycle of their container deployments as per the AWS Shared Responsibility Model. Methods of security mitigation outlined within this whitepaper will help customers to address PCI DSS compliance requirements for their containerized workloads.

Contributors

The following individuals and organizations contributed to this document:

- Arindam Chatterji, Sr. Solutions Architect, AWS
- Tim Sills, Sr. Solutions Architect, AWS

Document revisions

Date	Description
June 25, 2021	First publication