The **deployment of containers** (What) is fully automated through a **centralized GitHub repository** (Where), which is maintained and managed by the **NEF teams** (Who). The control ensures that only the **centralized team** (Who) can modify the security configurations of deployed containers, and these configurations are based on **CIS industry security standards** (What). The control is implemented **to prevent unauthorized changes to container security configurations** (Why) by enforcing strict validation of container images against predefined security baselines during the CI/CD pipeline process. It is **consistently applied during every deployment** (When), with no modifications allowed by the self-serve team during or after the deployment (How). The system **rejects any container images or configurations that fail to meet these standards**, ensuring **100% compliance with security requirements** (How much).

**How is the Control Performed?**

**Describe the control key steps at a high level:**

**Inputs and Sources:**

* Container images developed by the NABserv team.
* Centralized GitHub repository used for deployment automation.
* Predefined security configurations based on CIS industry security standards.

**What is done to process the inputs:**

* Container images are validated against security baselines based on CIS industry standards within the CI/CD pipeline.
* Centralized tools enforce security configurations to ensure compliance, ensuring that only authorized configurations are applied to the deployed containers.
* Only the centralized team can make changes to the security configurations of deployed containers; no modifications are allowed by the self-serve team.
* The CI/CD pipeline automatically rejects any container images or configurations that do not meet the predefined security baselines.

**Relevant thresholds/limits:**

* Security configurations must align with CIS industry security standards, with no critical vulnerabilities or misconfigurations.
* Only the centralized team has the authority to modify security configurations; changes by the self-serve team are prohibited.
* Post-deployment configuration changes are strictly prohibited.

**How outliers are tracked, managed, escalated:**

* Non-compliant security configurations are flagged by the centralized tools and alerts are generated.
* Alerts for unauthorized configuration changes or violations are escalated to the centralized NABserv team for investigation and remediation.

**How the control review is evidenced:**

* Deployment logs from GitHub provide a record of control enforcement, including the acceptance or rejection of containers.
* Reports from security tools demonstrate compliance with CIS industry security standards.
* Audit trails from the centralized tools show evidence that the deployed configurations adhere to security baselines and that no unauthorized changes were made post-deployment.

**Control Description (5W2H)**

* **What:** The deployment of containers is automated through a centralized GitHub repository, with the security configurations based on CIS industry security standards.
* **Why:** The control ensures that only the centralized team can modify the security configurations of the deployed containers, preventing unauthorized changes and ensuring compliance with security baselines.
* **Who:** The control is performed by the **NEF teams**, specifically the centralized team responsible for container deployment and security management.
* **When:** The control is applied consistently during every stage of the deployment process, and changes are restricted during and after the deployment.
* **Where:** The control is performed in the **centralized GitHub repository** used for the deployment of containers, and the security configurations are managed through automated tools.
* **How:** Automated tools integrated within the CI/CD pipeline validate container images and configurations against predefined security baselines. The pipeline rejects any non-compliant configurations and prevents modifications by non-authorized teams.
* **How much:** The control ensures **100% compliance** with predefined security baselines and prevents unauthorized changes to deployed containers.

**Control Objective**

* The control is performed **to prevent unauthorized modifications** to the security configurations of deployed containers, ensuring that all containers comply with predefined **CIS industry security standards**. It manages the issue by enforcing strict validation and rejection of non-compliant configurations, thus maintaining the integrity and security of deployed containers.

**Control Frequency**

* The control is performed **each time a container is deployed** through the CI/CD pipeline, ensuring continuous enforcement of security configurations during every deployment process.

**Control Ownership**

* The control is **performed by the NEF teams**, who are responsible for maintaining the centralized GitHub repository and managing the security configurations. The **centralized team** is accountable for ensuring the security baselines are followed and no unauthorized changes occur during or after deployment.

**Control Type**

* The control is **preventative**, as it prevents unauthorized changes to container configurations during deployment and post-deployment.

**Control Classification**

* The control is **automated**, as it relies on tools integrated into the CI/CD pipeline to enforce security standards and automatically reject non-compliant configurations.

**Where is it performed**

* The control is performed in the **centralized GitHub repository** and the associated **CI/CD pipeline environments**, where all container deployments and security configurations are managed.

**Background Detail**

* The control is critical in ensuring that the deployment process adheres to **industry-standard security practices** by leveraging **CIS security baselines**. It restricts unauthorized access to security configuration changes and maintains the integrity of the deployed containers by enforcing a standardized and secure deployment process.

**Impact Statement (Brief):**

1. **Security Vulnerabilities:** Inconsistent security controls expose the organization to potential misconfigurations and unauthorized access.
2. **Increased Exploitation Risk:** Asset teams' unrestricted modification of container images can introduce vulnerabilities, making systems more vulnerable to attacks.
3. **Compromised Security Posture:** Lack of centralized security enforcement for ECS and VM deployments weakens overall container security.
4. **Potential Data Breaches:** Inadequate security controls could lead to data breaches, regulatory non-compliance, and violations of security standards.
5. **Operational Disruptions:** Vulnerabilities in container configurations could cause system downtime and negatively impact business operations.

**Overall Control Test Result:**

**Fail** – The control did not ensure consistent enforcement of security practices across all container deployments within the organization. While the security configurations for Kubernetes container deployments (EKS and AKS) are successfully managed and protected by OPA policies and Gatekeeper, preventing unauthorized modifications, **non-Kubernetes container deployments (ECS and VMs)** lack any centralized security configuration or enforcement mechanism. Asset teams can freely modify container images and configurations without adhering to CIS security standards or using the approved CI/CD pipeline.

This inconsistency in security practices across different deployment methods results in a failure of the control, as there is no universal enforcement of security policies and standards for all containers deployed across the organization.

**Control Assessment**

|  |  |  |
| --- | --- | --- |
| **Design Effectiveness Assessment Questions** | **Result** | **Rationale** |
| **Is this the right control to mitigate the risk?** | **Fail** | The control does not effectively mitigate the risk for non-Kubernetes container deployments (ECS and VMs), as they are not managed under the same security framework. |
| **Does the control achieve its objective?** | **Fail** | The control fails to achieve its objective as non-Kubernetes deployments (ECS and VMs) lack security configuration management, resulting in inconsistent security practices. |
| **Is the control performed by the right people with requisite skills, knowledge and experience?** | **Fail** | Non-Kubernetes deployments are not managed by the right teams with the requisite expertise, as asset teams have the ability to modify container configurations. |
| **Does the control have adequate segregation of duties?** | **Fail** | The control does not have adequate segregation of duties for non-Kubernetes deployments, as asset teams can modify container configurations without oversight or restrictions. |
| **Is the control performed at the right time or in the right stage of the process?** | **Fail** | The control is not applied consistently across all container deployments, with no enforcement for non-Kubernetes deployments during critical stages of the container lifecycle. |
| **Is the control performed at the right frequency?** | **Fail** | The control is not implemented enterprise-wide for all containers. Non-Kubernetes deployments lack a security framework to ensure ongoing enforcement across the deployment pipeline. |
| **Is the control sustainable?** | **Fail** | The control is not sustainable for all container deployment methods. The absence of centralized security tools for non-Kubernetes deployments undermines long-term effectiveness. |
| **How does it manage/escalate an issue?** | **Fail** | There are no security tools in place for non-Kubernetes deployments to detect and manage security misconfigurations, leading to undetected risks and issues. |
| **Is the control evidenced?** | **Pass** | The control was validated through tests performed during the audit, specifically for Kubernetes deployments, which showed compliance with security configurations. |
| **Does the control have adequate management focus?** | **Pass** | Management focus is on shifting towards Kubernetes-based container deployments, with a greater emphasis on improving security controls for Kubernetes services. |

**Exceptions:**

* **Exception 1 Raised:** "The control failed for non-Kubernetes deployments (ECS and VMs), as there are no enforced security standards or centralized tools to manage configurations, leaving these deployments vulnerable to modification by asset teams."

The control attribute being tested is **security configuration management for container deployments**.

The test verifies that the security configuration of containers cannot be modified by unauthorized users, ensuring centralized control over container security settings across all deployment methods.

Audit concludes that this test has failed due to the lack of centralized security configuration management for non-Kubernetes container deployments. While Kubernetes deployments (EKS/AKS) are effectively secured through OPA policies and Gatekeeper, preventing unauthorized modifications, non-Kubernetes container deployments (such as those using ECS or VMs) do not have similar enforcement mechanisms in place. As a result, asset teams have the ability to modify security configurations and container images without restriction, which exposes the organization to potential security vulnerabilities and non-compliance with established security standards.

**Control Test: Kubernetes and Non-Kubernetes Container Deployment Security Configuration**

**Test Step:**

* **Test 1: Kubernetes Deployment Security Configuration**
  + Verify that container images deployed in Kubernetes clusters (EKS and AKS) cannot be modified by asset teams by changing the security configuration of containers.
  + **Test Step Details:**
    - Evaluate security configuration management for Kubernetes deployments via OPA policies and Gatekeeper, ensuring that asset teams cannot modify security configurations, including privileged escalation settings or root user access, for containers.
    - Check policies set within the kube-system namespace, which restrict access to security settings, and ensure that only the NEF team can modify these settings.
    - **Evidence Reviewed:** Audit logs, Gatekeeper policies, and access logs for security configurations.
  + **Test Execution:**
    - Modify the "allow privilege escalation" and "root user access" parameters for container images in the Kubernetes cluster and verify that these changes cannot be executed due to the restrictive policies.

**Outcome:**

* + The security configuration for containers deployed via Kubernetes (EKS and AKS) cannot be modified due to the active OPA policies and Gatekeeper restrictions, as expected.
  + **Result:** The control is effective for Kubernetes container deployment.
* **Test 2: Non-Kubernetes Deployment Security Configuration**
  + Verify the security configurations and control mechanisms for containers deployed through ECS and Virtual Machines (VMs) and assess whether there are any standards for enforcing CIS security configurations.
  + **Test Step Details:**
    - Review whether ECS or VM-based container deployments have any enforced security policies in place (such as those used in Kubernetes).
    - Verify if asset teams can modify container images or security configurations without restrictions and whether these actions are monitored or controlled centrally.
    - **Evidence Reviewed:** ECS task definitions, VM configuration files, deployment logs.
  + **Test Execution:**
    - Check if asset teams can modify container images or security configurations freely without enforcement of security policies or use of the enterprise-approved CI/CD pipeline.

**Outcome:**

* + No standard security configuration, tools, or policies are in place for container deployments in ECS or VMs. Asset teams have unrestricted ability to modify container images, and there are no centralized security configurations applied to these deployments.
  + **Result:** The control fails for non-Kubernetes container deployments.

**Evaluation Criteria:**

* **Kubernetes Deployment:** The test checks whether the security configurations (privilege escalation, root access) can be modified by non-authorized users. The criteria are based on OPA policies and Gatekeeper configurations as per security guidelines.
* **Non-Kubernetes Deployment:** The test checks for the presence of security policies or tools to enforce CIS security standards in ECS and VM environments. The criteria are based on ensuring that security configurations are managed centrally and consistently across all container deployments.

**Test Execution Summary:**

* **Kubernetes Deployment:**
  + Sub-sample: Randomly selected Kubernetes containers deployed via EKS and AKS clusters.
  + Reviewed: GitHub repository configurations, OPA policies, and Gatekeeper settings in the kube-system namespace.
  + Execution: Attempted to modify security parameters for containers, and found modifications were prevented due to Gatekeeper policies.
* **Non-Kubernetes Deployment:**
  + Sub-sample: ECS and VM deployments across multiple asset teams.
  + Reviewed: ECS task definitions, VM configuration files, and deployment logs.
  + Execution: Found no enforcement of security configurations or policies for ECS/VM container deployments, and asset teams could freely modify configurations.

**Control Test Result:**

* **Pass for Kubernetes deployments** as the control successfully prevents unauthorized modifications.
* **Fail for non-Kubernetes deployments** as there is no centralized security configuration or enforcement mechanism, and the asset team has the ability to modify security settings.

**Exceptions:**

* **Exception 1 Raised:** "IA found that no security configuration or tool is in place to maintain CIS security standards for ECS or VM-based container deployments, and asset teams can modify the container images and configurations without restriction."