**Test 1: Containers Network Security Configuration Cannot Be Modified by Asset Team**

**Control**

The NEF team centrally manages the configuration of network policies, ensuring that containers' network security configurations cannot be modified by the asset team.

**Control Test**

**Evaluation Criteria:**

* The test ensures that network policies and security configurations are centrally managed, cannot be altered by the asset team, and are enforced per security standards and organizational policy.
* Criteria are defined in NEF's Standard Operating Procedures for centralized network policy management, which are reasonable to mitigate risks of unauthorized changes.

**Key Test Steps:**

1. **Sampling and Scope:**
   * Sub-sampled Kubernetes-based services (EKS and AKS) and non-Kubernetes services (ECS and virtual machines) for evaluation.
   * Referenced evidence files, GitHub repositories, and policy logs:
     + GitHub Repo: NEF/tenants (EKS configuration).
     + GitHub Repo: nef-2.0-aks-admin (AKS configuration).
2. **Execution:**
   * Reviewed EKS and AKS network configurations to validate segmentation policies and enforcement by the NEF team.
   * Tested communication between pods within the same namespace for EKS (e.g., ms-nabone-accounts-sit1 and ms-nabone-accounts-sit4) to ensure segmentation.
   * Evaluated security group management in ECS and virtual machines to verify whether asset teams could modify configurations.

**Outcome:**

1. **Kubernetes-Based Services (EKS and AKS):**
   * **EKS Results:**
     + Separate landing zones with distinct CIDR ranges validated.
     + Network segmentation enforced through VPCs, security groups, and forward proxy settings.
     + Audit testing confirmed that communication between pods within the same namespace (ms-nabone-accounts-sit1 and sit4) is blocked via forward proxy.
     + No issues noted.
   * **AKS Results:**
     + Network policies centrally managed via GitHub (nef-2.0-aks-admin).
     + Policies restrict ingress/egress traffic between namespaces, ensuring no unauthorized communication.
     + Audit confirmed communication controls in place for acq-prod namespace.
     + No issues noted.
2. **Non-Kubernetes Services (ECS and Virtual Machines):**
   * Security groups for ECS and virtual machines can be modified by the asset team.
   * Testing revealed no network policies to restrict container communication.
   * Fail – Containers within ECS and virtual machines can communicate freely, breaching segmentation controls.

**Comparison Against Expectations:**

* **Kubernetes Services:** Passed as expected. Configurations align with security standards and are centrally managed.
* **Non-Kubernetes Services:** Failed as security groups can be modified, and communication controls are ineffective.

**Control Test: Internet Accessibility from Containers**

**Control**

The control ensures that containers do not have unnecessary internet accessibility, thereby restricting access to public repositories and mitigating risks associated with downloading unauthorized code or packages.

**Control Test**

**Evaluation Criteria:**

* The test ensures that internet access from containers is restricted per the organization's security policies and procedures.
* Criteria are defined in NEF's Standard Operating Procedures for container network security, which require restricted internet access to prevent the downloading of unauthorized external content.

**Key Test Steps:**

1. **Sampling and Scope:**
   * Sub-sampled Kubernetes services (EKS and AKS) and non-Kubernetes services (VMs).
   * Accessibility to public GitHub repositories was tested across these environments.
   * Evidence reviewed: Screenshots and logs showing internet access and downloaded files.
2. **Execution:**
   * **From Kubernetes Service EKS:**
     + Verified accessibility to the public GitHub Helm repository.
     + Observed that non-root users could download external files from the repository and manipulate them.
   * **From AKS:**
     + Tested accessibility to a public GitHub test repository.
     + Confirmed non-root users could download and work on files from the repository.
   * **From Non-Kubernetes Service (VM):**
     + Tested with the Murex application asset team.
     + Confirmed that files from public GitHub repositories could be downloaded and executed with root access in VMs.

**Outcome:**

* Internet accessibility was observed in all tested environments (EKS, AKS, and VMs).
* Containers could download files from public GitHub repositories.
* This capability allowed non-root and root users to access and manipulate external files, potentially exposing the environment to unauthorized or malicious code.

**Comparison Against Expectations:**

* **Expected Outcome:** Internet access should be strictly restricted to prevent unauthorized downloads and ensure secure container management.
* **Observed Outcome:** Internet access was available via forward proxy, allowing containers to download files from public GitHub repositories.
* **Gap:** The level of restriction provided by the forward proxy was more permissive than needed for container management.

**Failure Reason and Impact:**  
**Control Attribute:** Internet accessibility from containers is restricted to prevent unauthorized downloads or access to public repositories.

**Failure Reason:**

* Containers hosted in Kubernetes (EKS and AKS) and non-Kubernetes services (VMs) could access public GitHub repositories via a forward proxy.
* The forward proxy provided restricted but overly permissive internet access, allowing non-root and root users to download external files.
* Unauthorized access to external content increases the risk of downloading malicious packages or running unauthorized scripts, which could compromise container security and integrity.

**Impact:**

* **Security Risk:** Unauthorized access to public repositories allows the possibility of introducing malicious code or packages into the container environment.
* **Operational Risk:** Uncontrolled downloads could lead to performance issues or conflicts within containerized applications.
* **Compliance Breach:** This failure violates organizational security policies that mandate restricted internet access for containers.
* **Increased Attack Surface:** Containers with internet access can become an entry point for attackers to exploit vulnerabilities.

**Rationale for Test Result:**  
The control failed because it did not effectively restrict internet access, allowing containers to download unauthorized external files, which introduces potential security vulnerabilities.

**Control Test Result**

**Fail** – The control did not effectively restrict internet access from containers, allowing unauthorized downloads and introducing potential security vulnerabilities.

**Exceptions**

**Exception 1:** Containers in EKS, AKS, and VMs were found to have restricted but permissive internet access, allowing them to access public repositories and download external content, contrary to organizational security standards.

**Design Effectiveness Assessment Table Based on Test 1 and Test 2**

|  |  |  |
| --- | --- | --- |
| **Design Effectiveness Assessment Questions** | **Result** | **Rationale** |
| **Is this the right control to mitigate the risk?** | **Fail** | While the control is designed to mitigate the risk of unauthorized access and communication, it fails to fully address these risks in non-Kubernetes environments (ECS and virtual machines). Test 2 also reveals the control does not restrict unnecessary internet access. **Linked Exception: Test 1 and Test 2 Failures.** |
| **Does the control achieve its objective?** | **Fail** | The control achieves its objective for Kubernetes-based services (EKS and AKS) but fails in non-Kubernetes environments (ECS and VMs). Additionally, internet access remains more permissive than required. **Linked Exception: Test 1 and Test 2 Failures.** |
| **Is the control performed by the right people with requisite skills, knowledge, and experience?** | **Pass** | The NEF team is responsible for implementing and managing network policies for Kubernetes services, demonstrating sufficient expertise. |
| **Does the control have adequate segregation of duties?** | **Pass** | Segregation of duties is maintained for Kubernetes-based services (EKS and AKS), with centralized management ensuring that asset teams cannot modify configurations. |
| **Is the control performed at the right time or in the right stage of the process?** | **Pass** | The control is applied during the deployment and management phases of Kubernetes services, ensuring consistent policy enforcement. |
| **Is the control performed at the right frequency?** | **Pass** | Network policies are centrally enforced and applied consistently in Kubernetes services. |
| **Is the control sustainable?** | **Fail** | The control is sustainable for Kubernetes-based environments but not for non-Kubernetes environments, where modifiable security groups and the lack of network policies undermine sustainability. **Linked Exception: Test 1 Failure (Non-Kubernetes Services).** |
| **How does it manage/escalate an issue?** | **Fail** | Issues related to internet access and modifiable security groups in non-Kubernetes environments are not effectively escalated or managed. **Linked Exception: Test 1 and Test 2 Failures.** |
| **Is the control evidenced?** | **Pass** | The control for Kubernetes services is well-documented and evidenced through GitHub repositories and audit logs. |
| **Does the control have adequate management focus?** | **Pass** | Kubernetes-based services receive adequate management focus with centralized control. However, non-Kubernetes services lack equivalent focus, leading to identified gaps. |

**Summary**

The control demonstrates effectiveness and sustainability in Kubernetes environments (EKS and AKS) but fails in non-Kubernetes services and in restricting internet access. The linked exceptions highlight areas needing remediation to strengthen overall control design.

**Rationale for Test Result:**

* **Kubernetes Services:** The control is effective as policies are centrally managed, and segmentation controls work as intended.
* **Non-Kubernetes Services:** The control is ineffective for ECS and virtual machines due to modifiable security groups and lack of network policies.

**Control Test Result**

* **Kubernetes Services:** **Pass** – The control ensures network security configurations are centrally managed and effective.
* **Non-Kubernetes Services:** **Fail** – The control did not restrict container communication due to modifiable security groups.

**Exceptions**

* **Exception 1:** Non-Kubernetes services (ECS and virtual machines) lack network policies, allowing unrestricted container communication.