Description

NAB utilizes Amazon EKS and AKS as centralised container orchestration services to host and manage key banking applications and databases. These services use the centrally managed NEF2 framework to ensure consistent security and operational standards. NAB also leverages Amazon ECS  for containerised application hosting and management using NEF1 framework. Unlike EKS and AKS, ECS follows a decentralised model where asset teams manage their own environments through CI/CD pipelines. Asset teams also have the capability to deploy containers directly onto EC2 instances in AWS and Azure VMs in Azure cloud. Audit identified the following gaps in the overall management of container lifecycle at NAB.

Design and Implementation phase

1 – Management did not define minimum security configuration baseline for containers (mBSS is not defined) Make it active voice

There is a lack of a formal minimum security configuration baseline document for containers. Audit further noted that work is in progress to create a Kubernetes technical specification document which will ensure NAB approved minimum security configuration baseline for containers managed using Kubernetes like Amazon EKS and AKS. There is no such document available for non-Kubernetes based containers like Amazon ECS or containers running on Amazon EC2 or Azure VM directly.

2 – Management inconsistently and inappropriately implemented security configurations

Containers running in the AWS environment

Audit sampled critical banking applications where containers are directly running on EC2 instance (Murex) and containers running on both EKS and ECS (NabTrade, Internet Banking, and JBWere) and identified the following misconfigurations assessed as per best practices recommended by global standard, Center for Internet Security (CIS):

**•   Privileged containers running directly on EC2 instances allowing host system access**

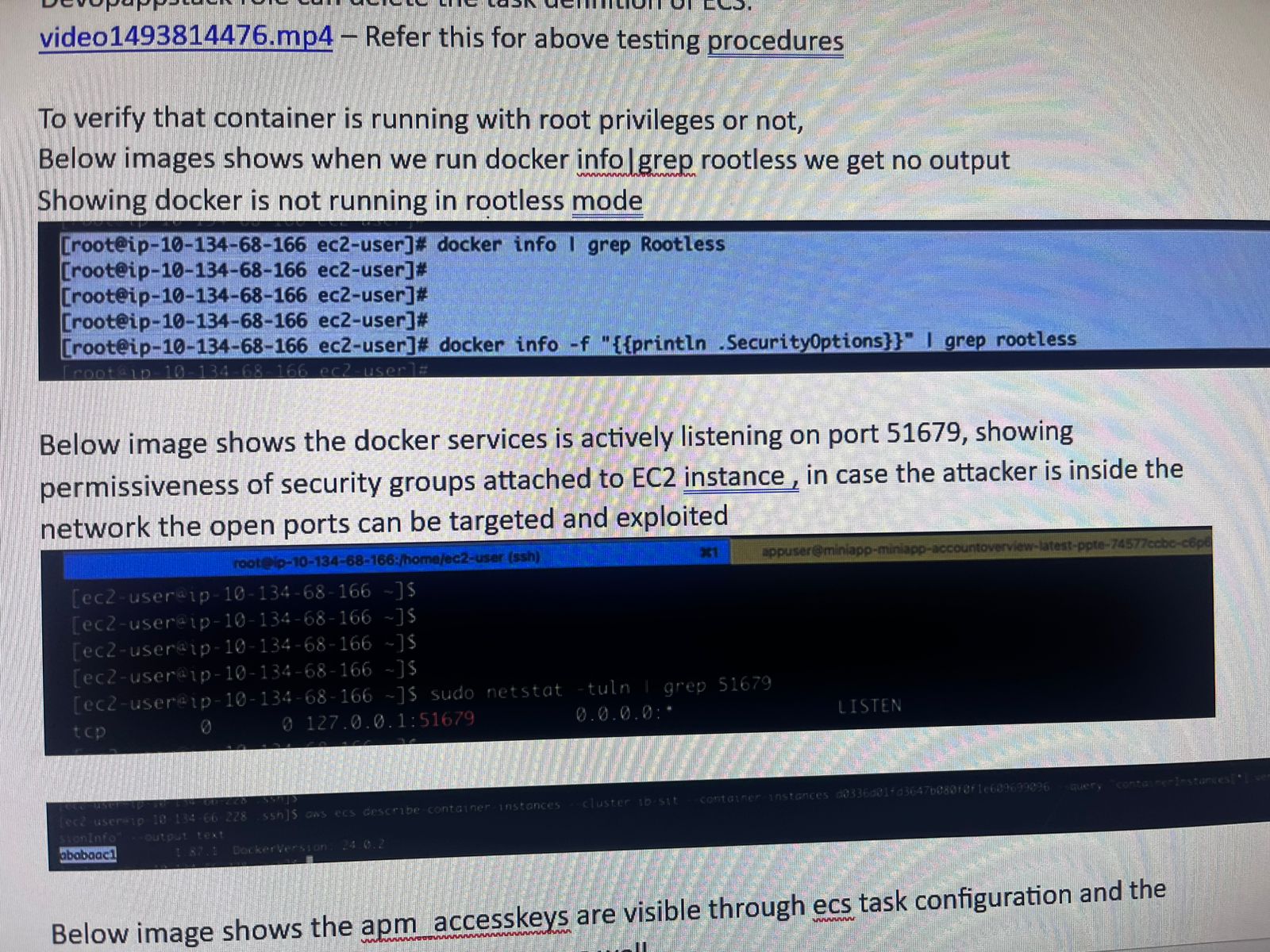
Containers running with privileged access on EC2 instances can directly interact with the host system using the highest privileged root access. This configuration increased the risk of security breaches, as it could lead to the compromise of the entire EC2 instance and all containers hosted on it, undermining the security and integrity of the environment.

Assets teams are able to SSH into virtual machines as root users, on which containers are running, with no security configurations in place to restrict this privileged access. These containers, whether deployed through the ECS service or directly running on virtual machines (EC2 instances), possess the capability to operate in privileged mode. This lack of control poses a security risk, as privileged containers can interact with the host system using root-level permissions. In the event of an internal breach, an attacker could exploit these capabilities to gain root access to the virtual machine, leading to the compromise of the entire EC2 instance and all containers hosted on it.

**Implications:** There are no security configurations enforced to restrict privileged access for containers. Consequently, privileged containers can interact with the host system with the highest root access, increasing the risk of security breaches. This vulnerability could lead to the compromise of the entire EC2 instance and all containers hosted on it, thereby undermining the security and integrity of the environment.

**Risk Scenario:** In the event of an attack, an attacker within the environment could exploit the privilege escalation capabilities of a container, thereby compromising the container and potentially gaining root access to the host virtual machine. This situation could facilitate lateral movement within the network  
**Attack Scenario:**

1. **Initial Compromise**: An attacker gains access to the container running with privileged access, possibly through a vulnerability in the application or misconfiguration.
2. **Privilege Escalation**: The attacker exploits the privileged container to gain root access on the host EC2 instance.
3. **Lateral Movement**: With root access, the attacker can move laterally within the internal network, potentially compromising other containers and services.
4. **Persistence**: The attacker installs backdoors or rootkits to maintain access even if the initial entry point is discovered and mitigated.(this is possible we can download the files



1. **Docker Not Running in Rootless Mode**:
   * Output: docker info | grep Rootless
   * Docker is not running in rootless mode, which means it has higher privileges and more access to the host system.
2. **Open Port**:
   * Output: sudo netstat -tuln | grep 51679
   * Docker services are listening on port 51679.

**Worsening Scenario:**

If Docker is not running in rootless mode and is listening on open ports, this can significantly exacerbate security risks:

1. **Increased Attack Surface**: Running Docker with higher privileges (not in rootless mode) provides attackers with more potential entry points and greater access if they manage to compromise a container.
2. **Network Exposure**: Open ports, especially if improperly secured, can be exploited by attackers to gain access to container services. Even if public access is restricted, an attacker within the network can exploit these open ports to launch attacks.
3. **Privilege Escalation**: With Docker not in rootless mode, an attacker compromising a container could escalate their privileges to gain control over the host system.
4. **Persistent Threats**: Attackers can install backdoors or rootkits on the host, maintaining access even if the initial vulnerability is patched.

**Technical Risk:**

* **Host System Compromise**: An attacker could gain root access to the host system, compromising the integrity of the entire environment.
* **Lateral Movement**: Attackers could move laterally across the network, targeting other containers and systems.
* **Data Exfiltration**: Sensitive data could be accessed and exfiltrated by attackers.
* **Service Disruption**: Attackers could disrupt services by exploiting open ports and gaining elevated access.

This scenario highlights the importance of running Docker in rootless mode where possible, securing open ports, and enforcing strict security configurations to mitigate potential risks.

The audit has revealed that users are able to SSH into virtual machines as root users, on which containers are running, with no security configurations in place to restrict this privileged access. These containers, whether deployed through the ECS service or directly running on virtual machines (EC2 instances), possess the capability to operate in privileged mode. This lack of control poses a significant security risk, as privileged containers can interact with the host system using root-level permissions.

Additionally, it has been observed that Docker is not running in rootless mode and services are listening on open ports, further exacerbating security risks. Running Docker with higher privileges and having open ports significantly increase the attack surface. In the event of an internal breach, an attacker could exploit these capabilities to gain root access to the virtual machine. They could move laterally within the internal network, compromising other containers and services, and exfiltrate sensitive data. This could undermine the security and integrity of the environment, facilitating persistent threats and severe operational impacts.

•   Periodic patching was not enforced for containers running on EC2 instances

Containers running directly on EC2 instances relied on individual asset teams for timely patching, leading to inconsistent practices. Currently, there was no centralised control or process to ensure timely patching of the container environment.

The audit observed that there is no periodic patching of the containers running on ECS and directly on virtual machines (VMs). This issue is highlighted by the fact that some Docker images have not been patched for up to three years, indicating a significant GIRP (Governance, Risk, and Compliance) violation. The lack of regular patching poses a significant security risk, as outdated containers are more vulnerable to exploitation due to unpatched vulnerabilities. This could lead to potential security breaches, including unauthorized access, data exfiltration, and service disruptions. In an attack scenario, an attacker could exploit vulnerabilities in outdated containers to gain access to sensitive information or compromise the entire environment. The absence of periodic patching allows attackers to leverage known vulnerabilities to escalate privileges, move laterally within the network, and maintain persistent access, thereby compromising the security and integrity of the organization’s systems.

**Periodic Patching of Containers**

**Finding:** Audit observed that there is no periodic patching of the containers running on ECS and directly on virtual machines (VMs). The attached image highlights the issue, showing Docker images that have not been patched for up to three years, indicating a significant GIRP (Governance, Risk, and Compliance) violation.

**Implications:** The lack of regular patching poses a significant security risk, as outdated containers are more vulnerable to exploitation due to unpatched vulnerabilities. This could lead to potential security breaches, including unauthorized access, data exfiltration, and service disruptions.

**Risk Scenario:** In an attack scenario, an attacker could exploit vulnerabilities in outdated containers to gain access to sensitive information or compromise the entire environment. The absence of periodic patching allows attackers to leverage known vulnerabilities to escalate privileges, move laterally within the network, and maintain persistent access, thereby compromising the security and integrity of the organization’s systems.

•   Lack of preventative controls to ensure network isolation of containers

Containers running on ECS and directly on EC2 instances can communicate with each other without restrictions. The network isolation for these containers was managed at host level and did not provide granular isolation at the container level. This allowed unnecessary inter-container communication, which expanded the attack surface and increased the potential for unauthorised access to sensitive resources if container roles or purposes were not properly segmented. This also created a challenge to implement of Zero Trust networking principles which required strict controls over which components of an application could communicate.

Internal Attack Vector: A malicious insider, compromised application, or unauthorized script on one container gains access to the network.

Lateral Movement:Due to inadequate isolation, the compromised container communicates freely with other containers or EC2 instances. The attacker scans and identifies containers or instances running sensitive workloads or holding privileged IAM roles.

Unauthorized Access:The attacker exploits inter-container communication to access sensitive data, steal IAM credentials, or disrupt workloads.

Impact:The attack compromises sensitive resources, disrupts containerized services, and potentially escalates privileges to other AWS resources or environments.

•   Inadequate protection of secret keys stored in AWS Secrets Manager from unauthorised access

Audit noted that the secret keys used by application monitoring tool (appd) for containerised applications were inadequately protected as secret values are visible to the users having privileged Identity and Access Management (IAM) role, DevopsAppstack. This introduced the risk of secret value exposure and violated the least privilege principle. If the container was compromised, an attacker could extract these secrets and gain unauthorised access to sensitive resources like databases or application code.

**Scenario of Exploitation**

1. **Initial Access:**
   * A privileged user with the **DevOpsAppstack** role accidentally exposes or intentionally shares their credentials.
   * Alternatively, a compromised container retrieves the secret from AWS Secrets Manager.
2. **Secret Exposure:**
   * The exposed secret allows the attacker to access critical resources such as application databases, sensitive APIs, or application code repositories.
3. **Unauthorized Access and Impact:**
   * Using the compromised secret, the attacker:
     + Extracts sensitive data from databases.
     + Modifies or injects malicious code into applications.
     + Executes privilege escalation by leveraging additional compromised resources.
4. **Broader Consequences:**
   * Service disruptions, data breaches, or financial loss occur due to the compromise of sensitive resources.

•   Unauthorised access to sensitive server metadata increased the risk of cyber attacks

EC2 instances hosting containers were not configured to enforce Instance Meta Data Service v2 (IMDSv2) which is a session-oriented method to access EC2 metadata. Audit noted that the less secure IMDSv1 can be enabled, which lacked session authentication. This exposed the instance metadata service to potential exploitation, such as Server-Side Request Forgery (SSRF) attacks, allowing attackers to retrieve sensitive metadata, including IAM role credentials.

**Attack Vector: Server-Side Request Forgery (SSRF)**

* **What Happens:** An attacker leverages a vulnerability in an application running on the EC2 instance to send unauthorized HTTP requests to the metadata service (available at http://169.254.169.254/).
* **Why It Works with IMDSv1:** IMDSv1 lacks session-based authentication or other security mechanisms, allowing any application or script on the instance to access the metadata if the request is crafted properly.

•   More permissible internet access allowed in the containers

Audit also observed that internet was accessible from the containers via forward proxy. This gave restricted internet access in the container but was more permissive than needed for container management. For example, www.google.com was accessible, which was not required in the containers. Internet access increases the attack surface.

Audit noted github external repository can be accessed, this gives the capability to run the open source code and packages over the containers with the help of virtual machine root privileges.

For the container that do not have root access then packages can run using non-root users can be used to exploit the vulnerability.

**Risk:** **Excessive Internet Access in Containers**

* The audit found that containers had unrestricted internet access via a forward proxy, which was more permissive than necessary for container management. This exposed the containers to unnecessary risk, as the containers could access external sites like [www.google.com](http://www.google.com/) and other unnecessary domains.
* **Relevance to the Audit:** The internet access allowed in containers increases the attack surface. Access to external websites could allow malicious code to be downloaded, thereby compromising the security of the containerized environment. Additionally, it provides avenues for unauthorized communications or data exfiltration.
* **Example:** Access to repositories like **GitHub** means that malicious or unverified code could be downloaded, compiled, or run within the container, which could introduce vulnerabilities, especially if the container has root access.

**Risk:** **Potential Exploitation of External Repositories**

* + **Relevance to the Audit:** The audit observed that the containers could access **GitHub repositories** and external resources, allowing them to pull open-source code or packages. This capability could be exploited if unauthorized packages or scripts from external repositories are run within the containers. Additionally, this exposes the environment to vulnerabilities in the packages from these external sources.
  + **Example:** Running unverified code from external repositories could compromise the integrity of the system, especially if the code contains known vulnerabilities or malicious intent. Allowing internet access in the container means that even non-root users may download and execute malicious code.

Containers in the Azure environment

39,513 out of 331,077 (12%) containers in production and non-production were using AKS as per Crowdstrike EDR report dates 12 October 2024. Audit observed the following gaps for containers running on AKS:

•   Enabled public access to Application Programming Interface (API) server in the Azure cloud

This configuration would allow an unauthorised user, access to the API server which in turn may allow access to NAB assets in Azure. Out of 25 Kubernetes services, 6 were accessible publicly, exposing the cluster to external threats, unauthorised access, and denial of service (DOS) attacks in the event of a system compromise.

Likelihood to exploit this is very less, as authentication to API server is in place.

Exploitation scenario  
Discovery: Attackers locate the publicly accessible AKS API server through scanning tools, increasing visibility.

Credential Theft: Authentication can be bypassed if credentials are stolen via phishing, social engineering, or brute force attacks.

Exploitation of Vulnerabilities: Attackers exploit unpatched Kubernetes API server vulnerabilities to bypass authentication or escalate privileges.

Session Hijacking: If a legitimate user's session token is compromised, attackers can impersonate them to access sensitive resources.

•   Disabled host level encryption for the Azure VM Scale Set (VMSS) used for hosting containers

AKS VMSS does not support NAB approved Operating System (OS) images like Azure Machine Images (AMIs), i.e. VMs are running without NAB’s approved security baseline configuration. There was a lack of encryption implemented on the data drive  level, increasing the risk of unauthorised access to sensitive data in the event of a security breach or physical compromise of the underlying infrastructure. This was also a violation of the Cloud Security Assurance Methodology (CSAM) policy, which required Virtual Machines (VMs) or hosts deployed as part of VMSS must have encryption at host enabled.  
Scenario of Exploitation

Initial Compromise:In the event of a breach or physical compromise of the underlying infrastructure, an attacker could gain access to the VMSS instances that lack encryption.

Data Exposure:With no disk encryption, sensitive data stored on the VM disks (e.g., user credentials, application data, logs) can be accessed directly, potentially leading to data exfiltration.

Impact:Exposure of sensitive customer or internal data, violation of privacy regulations, and potential escalation of attacks to other parts of the cloud infrastructure.

Monitoring and reporting phase

3 – Management did not monitor the critical configurations in the container environment to ensure alignment with globally recognised baseline security requirements

Container security practices did not fully align with NAB baseline security requirements due to lack of monitoring and scanning of critical configurations in container images.

Lack of container security configuration scanning and monitoring

Critical configurations for container runtime were not consistently monitored for compliance with NAB baseline security requirements. Although Snyk security tool was monitoring the container environment, it was limited to the scanning of vulnerabilities and not for security configurations in the containers for compliance purpose.

Inconsistent scanning across platforms

Scanning was not mandatorily enforced for runtime configuration of containers running on Amazon ECS and VMs like Amazon EC2 and Azure VM.

Use of Non-NAB Approved OS Images for AKS

While non-NAB approved OS images for containers were deployed in AKS, the absence of critical configuration scanning increased the risk of undetected misconfigurations.

These gaps created potential vulnerabilities that could lead to security breaches in the container environment. Misconfigurations in the container environment could be exploited by malicious actors, resulting in unauthorised access, data breaches, or disruption of services.

4 – Management did not report on misconfiguration non-compliances appropriately

Crowdstrike EDR tool, managed by Group Security, performed container compliance assessment. Due to lack of centralised reporting on containers, Audit utilised the EDR dashboard as of 15 November 2024 and identified that container compliance failed for considerable number of containers against 20 rules (8 critical, 5 high, 5 medium and 2 low severity), as per globally accepted CIS standard. Some of the key failures across a population of 1,161,185 containers were as below:

•   (Critical) Ensure that the container is restricted from acquiring additional privileges – 60%

•   (Critical) Ensure sensitive host system directories are not mounted on containers – 24%

•   (Critical) Ensure that privilege containers are not used – 4%

•   (High) Ensure that the container's root filesystem is mounted as read only – 79%

•   (High) Ensure that Linux kernel capabilities are restricted within containers – 65%

Further, there was no reporting of critical misconfigurations in the container lifecycle management process for impacted assets up to the senior management including key Executives.

Impact

Failure to implement appropriate security configurations and timely detect and manage critical security gaps in the container environment could lead to significant financial losses due to potential data breaches, and operational disruptions. The configuration non-compliance, inadequate monitoring, and appropriate level of reporting up to the senior management could negatively impact systems driving customer services which in turn could erode customer trust, harm the NAB’s reputation.