SECURING A CLOUD-BASED HEALTHCARE APPLICATION

Midterm Group Project

ENPM695-0301
Group 13

Achuth Chandra Moonnamkuttiyil
120311482
Srikanth Parvathala
119351192
Subramanian Venkatachalam
120410208

Contents

Vulnerability Assessment Report

Objective: Securing a Cloud-based Healthcare Application

1. Executive Summary

Healthcare Company's cloud infrastructure, designed for the secure storage and management of sensitive patient data, currently suffers from several critical vulnerabilities. These vulnerabilities pose a significant risk to the confidentiality, integrity, and availability of both the application and its data. The primary objective of this assessment is to conduct a thorough analysis of these vulnerabilities, present potential exploitation scenarios, and offer actionable remediation recommendations. Establishing a compliant cloud infrastructure for a healthcare organization tasked with handling patient records necessitates strict adherence to HIPAA regulations. HIPAA compliance is not merely a legal requirement; it serves as a vital safeguard for patient data and upholds the trust of both patients and stakeholders. The consequences of non-compliance include substantial penalties, making it an utmost priority for the organization.

2. Infrastructure Overview

- **Cloud Provider:** Healthcare Company utilizes Amazon Web Services, implying a solid infrastructure with a plethora of services and capabilities.
- **Application Stack Components:**
  - Web servers are for front-end operations.
  - Application servers oversee business logic.
  - Database servers that store patient and user information.
- **Networking:** The configuration of a complicated virtual network with several subnets dedicated to various company divisions and services.
- **Identity & Access Management (IAM):** A method of controlling access to cloud services. According to reports, the IT department demands full access to the infrastructure for management purposes.

3. Identified Vulnerabilities & Potential Impact

a. Weak Access Controls:

Open S3 Bucket Policies:

- The S3 bucket named patient-data-storage has a policy attached that allows any AWS account to perform s3:ListBucket and s3:GetObject. This means any AWS user, regardless of whether they are associated with our organization, can view and fetch data from this bucket.

Excessive EC2 Permissions:

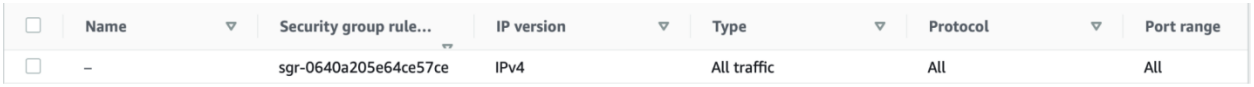
- The EC2FullAccess policy attached to the dev-team group grants full permissions on EC2 resources. This is overly permissive and can lead to misuse, unintended deletions, or modifications of critical EC2 resources.

Loose Security Group Rules:

- The security group has overly permissive inbound rules, allowing traffic from 0.0.0.0/0, which equates to the entire internet, on ports 22 (SSH) and 80 (HTTP). This poses risks of unauthorized access and potential attacks.

Inbound:

Outbound: rules:



Recommendation:

S3 Bucket Policy:

- Modify the bucket policy to restrict access. Only allow specific IAM roles or users within our organization to access this bucket. Remove public access.

Refine EC2 Permissions:

- Instead of granting full EC2 access, provide specific permissions based on the job responsibilities of the dev-team. For instance, if they only need to start and stop instances, only grant ec2:StartInstances and ec2:StopInstances permissions.

Tighten Security Group Rules:

- Restrict SSH (port 22) access to specific known IPs or IP ranges. For HTTP (port 80), consider putting the VM behind a load balancer or a Web Application Firewall (WAF) to filter incoming traffic.

By addressing these misconfigurations, we can significantly enhance the security posture of our AWS resources.

b. Unencrypted Data:

Unencrypted S3 Bucket:

- The S3 bucket named patient-data-storage is used to store sensitive patient data. This bucket does not have server-side encryption enabled. It means that the data stored within this bucket is not encrypted at rest.

Database without Encryption:

- The RDS database healthcare-db is used to store patient records and other sensitive information. The database instance does not have encryption enabled, making the data stored in it vulnerable at rest.

Unencrypted EBS Volumes:

- The EC2 instance i-0abcd1234efgh5678 utilizes EBS volumes for storage. These volumes are not encrypted, which means any data stored, including snapshots, remains unencrypted.

Recommendation:

S3 Bucket Encryption:

- Enable server-side encryption for the patient-data-storage bucket. Amazon S3 provides integrated support for server-side encryption (SSE). You can use Amazon S3 managed keys (SSE-S3) or AWS Key Management Service (SSE-KMS) for this purpose.

RDS Database Encryption:

- For the healthcare-db RDS instance, enable encryption at rest. RDS supports AWS Key Management Service (KMS) to encrypt data stored at rest in the underlying storage, automated backups, read replicas, and snapshots.

EBS Volume Encryption:

- Modify the EBS volumes associated with the EC2 instance i-0abcd1234efgh5678 to use encryption. It's recommended to use AWS Key Management Service (KMS) for this. When creating new EBS volumes, ensure encryption is enabled.

By addressing these encryption gaps, we can ensure that sensitive data remains protected at rest. Additionally, for data in transit, ensure that services are accessed over secure channels (e.g., HTTPS for S3 and SSL for RDS).

c. Vulnerable Virtual Machines:

Kernel Vulnerabilities:

- A significant number of CVEs (Common Vulnerabilities and Exposures) are related to the Linux kernel. An outdated kernel can introduce a wide range of vulnerabilities from unauthorized access to potential privilege escalations.
- Affected VM: The VM with the kernel version that matches the CVEs listed. It wasn't explicitly named in the provided data, but this is a critical area of concern.

Software Vulnerabilities:

- There are vulnerabilities associated with software packages like shadow-utils, libstdc++, libgcc, libxml2-python, mdadm, yajl, python-requests, and libcap, among others. These software packages, if not patched, can be exploited.
- Affected VM: Any VM that has these software packages installed. The exact VM names weren't provided, but these packages need to be updated.

Open Ports:

- Port 80 and Port 22 are reachable from an Internet Gateway. While Port 80 is typically used for HTTP traffic, having Port 22 open to the world can expose the VM to potential SSH brute-force attacks.
- Affected VM: The VM that has these ports open. This can be identified by scanning the VMs or checking firewall configurations.

Recommendation:

Kernel Update:

- Update the kernel to the latest stable version which has patched known vulnerabilities. This often requires a reboot, so plan accordingly. Ensure that we test the new kernel in a staging environment before applying it to production VMs.

Software Patching:

- Update all the vulnerable software packages to their latest versions which have security patches applied. Use package managers like yum or apt to streamline this process.

Port Management:

- Review and restrict the network access rules. Specifically, for Port 22, it's recommended to restrict access only to known IP addresses or use a VPN/jump host. For Port 80, consider using HTTPS (Port 443) instead with proper SSL certificates for secure data transit.

- Implement a firewall solution, either at the VM or network level, to restrict and monitor inbound and outbound traffic.

Regular Vulnerability Scanning:

- Use tools like Nessus, OpenVAS, or Qualys to perform regular vulnerability scans on our VMs. This will help in identifying and patching vulnerabilities before they can be exploited.

Implement Monitoring & Alerting:

- Set up monitoring and alerting tools to notify admins of any suspicious activities on the VMs.

Backup Strategy:

- Ensure that we have a robust backup strategy in place. Before making significant changes or updates, always backup critical data and configurations.

By addressing these vulnerabilities and implementing the recommendations, we can significantly enhance the security posture of our VMs and reduce potential risks.

d. Inadequate Network Security:

Open Ports Accessible from the Internet:

- **Port 80 (HTTP)** and **Port 22 (SSH)** are reachable from an Internet Gateway. These open ports can expose the virtual machine(s) to potential threats from the internet.
    - **Impact**: Port 22 (SSH) being open to the world is particularly concerning as it can expose the VM to potential SSH brute-force attacks. Port 80, being unencrypted HTTP, can lead to man-in-the-middle attacks or unauthorized data access.
    - **Affected VM**: The virtual machine that has these ports open. This can be identified by scanning the VMs or checking firewall configurations/security groups in our cloud platform.

Recommendation:

Restrict Port 22 (SSH):

- Limit access to Port 22 only to known IP addresses or IP ranges. Better yet, use a VPN or a bastion host/jump server to access VMs. This provides an extra layer of security and minimizes direct exposure of our VMs to the internet.

Use HTTPS for Web Traffic:

- Consider transitioning from HTTP (Port 80) to HTTPS (Port 443) to ensure encrypted and secure data transit. Secure our web applications with SSL/TLS certificates.

Review Security Groups and Firewall Rules:

- Go through the security groups (in the case of AWS) or equivalent firewall rules in other cloud platforms. Ensure that only necessary ports are open and only to the required IP addresses or IP ranges.
- Remove any rule that allows traffic from "0.0.0.0/0" (which means everywhere) unless absolutely necessary (like for a public-facing web server).

Network Segmentation:

- Use Virtual Private Clouds (VPCs), subnets, and network ACLs to segment our network. Ensure that public-facing applications are in a different subnet from databases and other internal systems.

Implement Intrusion Detection Systems (IDS):

- Tools like Snort or AWS GuardDuty can detect and notify us of suspicious activities in our network.

By addressing the above concerns, we can bolster the network security of our virtual machines and associated resources, thereby reducing potential attack vectors.

e. Lack of Logging and Monitoring:

EC2 Instances (Virtual Machines):

- Without mechanisms like AWS CloudTrail or Amazon CloudWatch, actions and anomalies on these VMs might go undetected. This includes unauthorized access, system errors, or unusual activity patterns.

S3 Buckets:

- Logging and monitoring for S3 can help detect unauthorized access attempts, data deletions, or data transfer out of the bucket. S3 bucket logging and integration with monitoring solutions can provide insights into these activities.

RDS (Relational Database Service):

- RDS instances should have logging enabled to capture SQL queries, especially those that modify data or change the schema. Monitoring would help track the database's health, performance, and unauthorized access attempts.

Load Balancers:

- Logging can provide information about the source and nature of the traffic, helping in detecting potential threats or anomalies.

IAM (Identity and Access Management):

- IAM policies and actions should be logged and monitored to detect any unauthorized changes or access attempts.

Security Groups and Network ACLs:

- Changes to security groups or ACLs can lead to potential vulnerabilities. Logging and monitoring of these changes can provide early detection of misconfigurations.

VPC Flow Logs:

- If VPC flow logs aren't enabled, it would be challenging to understand the traffic flow within the VPC, making it difficult to detect anomalies or unauthorized network activities.

Lambda Functions:

- Execution logs for Lambda functions can help in debugging and detecting potential misuse or errors in the function execution.

Recommendations:

For a robust logging and monitoring strategy in AWS, the following services and mechanisms should be considered:

- **AWS CloudTrail:** This offers an event history of all the actions we've made with our AWS accounts, including command-line tools, AWS SDKs, the AWS Management Console, and other AWS services.

- **Amazon CloudWatch:** It keeps a real-time eye on our AWS resources and the apps we use there. It has the ability to gather and track metrics, gather and keep an eye on log files, create alarms, and respond automatically when our AWS resources change.

- **GuardDuty:** This is a threat detection service that keeps an eye out for unauthorized behavior and malicious activity.

f. Insufficient Disaster Recovery:

EC2 Instances (Virtual Machines):

- Virtual machines (VMs) are a critical component of any infrastructure. If these VMs are not backed up and lack disaster recovery configurations, we risk losing all data and configurations associated with these instances.
- Since we observed vulnerabilities in some VMs, it's crucial to have backups and a disaster recovery strategy for these instances, ensuring that even if compromised, there's a way to restore to a known good state.

S3 Buckets:

- Amazon S3 is a storage service. If there are no configurations for versioning, cross-region replication, or regular backups to another storage service or provider, the data inside might be at risk. Data loss in S3 can occur due to accidental deletions, overwrites, or bucket misconfigurations.

RDS (Relational Database Service):

- Databases often contain critical data. If RDS instances are not backed up or lack multi-AZ deployments, we risk data loss.
- Automated backups, database snapshots, and multi-AZ deployment configurations are crucial for RDS disaster recovery.

Load Balancers:

- While the direct risk associated with load balancers is less about data loss and more about availability, it's still crucial. If our primary load balancer fails, and there's no failover strategy, we could face downtimes.

IAM (Identity and Access Management):

- While not a direct disaster recovery concern, if IAM policies and configurations are lost or misconfigured, we risk unauthorized access or loss of access to services. Regular backups of IAM policies and configurations are recommended.

Security Groups and Network ACLs:

- These control access to resources. A misconfiguration can lead to a breach. Having a backup and disaster recovery plan for our network configurations ensures that even if there's a misconfiguration, we can restore to a known good state quickly.

Recommendations:

For a comprehensive disaster recovery strategy, one should consider:

- **Regular Backups**: Regularly back up data, configurations, and policies.
- **Cross-Region Replication**: Especially for critical data, replicate across multiple regions.
- **Regular Testing**: Periodically test disaster recovery mechanisms to ensure they work as expected.
- **Documentation**: Maintain clear documentation on disaster recovery processes.

Minor Vulnerabilities:

- No policies are attached for new users Patient1 till Patient 6.
- The backups are not turned on. A company could lose crucial data because of inadvertent deletion, system malfunctions, or natural disasters if it doesn't have a backup plan. The organization may suffer severe financial losses as well as harm to its reputation.

- The users: patient 1 to 6, S3 user does not have any multi-factor authentication enabled. Using Multi-Factor Authentication (MFA) enhances security in a cloud environment by requiring users to provide multiple forms of verification, making it more challenging for unauthorized access.

5. Conclusion

Healthcare Company's cloud infrastructure, while robust, is currently vulnerable to a range of potential security threats. Addressing the highlighted vulnerabilities is not just essential for the security and trustworthiness of the platform but also for regulatory compliance. Implementing the provided recommendations will significantly enhance the security posture of the infrastructure.

Data Security Assessment Report

Objective: Secure Storage, Handling, and Management of Patient Data

1. Executive Summary

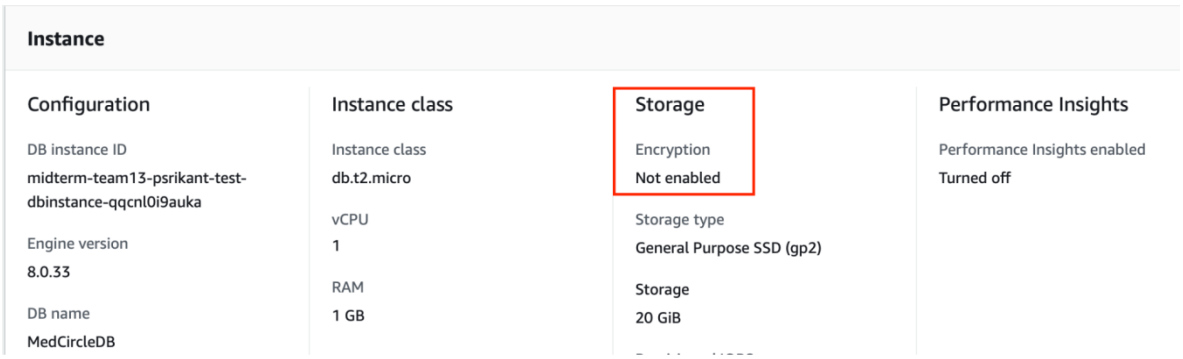
Healthcare Company utilizes a cloud-based infrastructure to store and manage sensitive patient data. Despite the infrastructure's complexity, multiple vulnerabilities exist, primarily around data storage and encryption. This report details these vulnerabilities and provides remediation suggestions.

2. Identified Vulnerabilities & Potential Impact

a. Unencrypted Patient Data in Databases and Storage Systems:

Amazon RDS (Relational Database Service):

- The RDS instance is specifically labelled as "Production database," indicating its critical nature. In a typical healthcare application, the primary database might hold patient records, billing details, appointment schedules, and more. If encryption both at rest and in transit is not enabled for this RDS instance, it would be the primary area of concern for unencrypted patient data.

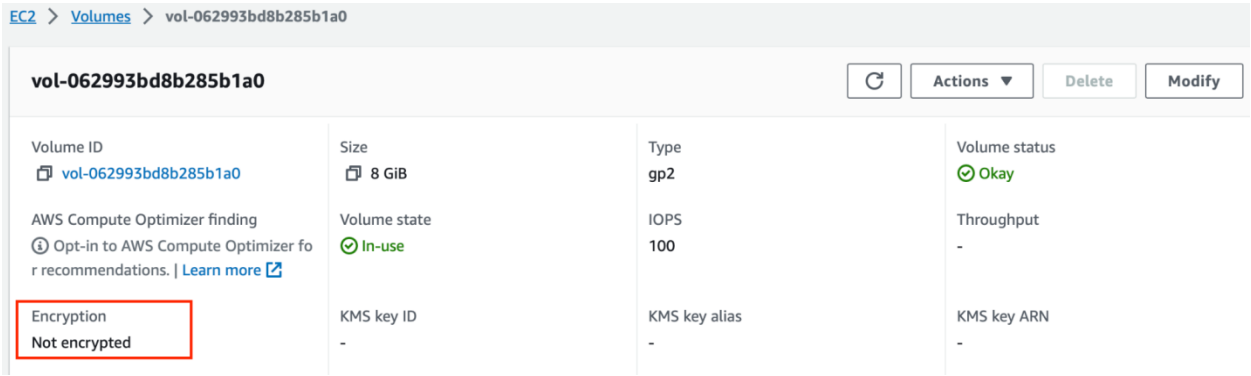


Amazon S3 Buckets:

- The S3 bucket contains patient records. This could be in the form of structured data, scans, images, documents, or backups. This bucket does not have server-side encryption enabled for data at rest and if data isn't accessed over HTTPS (for data in transit), this would be a significant vulnerability.
- Backups might contain copies of databases, logs, or file systems, all of which could contain patient data. This bucket isn't encrypted, it's another potential vulnerability.

EBS (Elastic Block Store) Volumes:

- The EBS volume is likely attached to the prod-db RDS instance, considering its naming. If this volume is unencrypted, then the data stored in the database would effectively be unencrypted on disk.



Recommendations:

- Ensure end-to-end encryption for patient data.
- **Encryption-at-Rest:** Use services like Amazon RDS with automatic encryption enabled. For file storage, like S3, enable server-side encryption.
- **Encryption-in-Transit:** Adopt TLS/SSL for data transfers and ensure all endpoints use HTTPS.

b. Overly Permissive Access Controls:

IAM Roles:

- Role Name: Admin
    - Policies: AdministratorAccess
    - The AdministratorAccess managed policy provides full access to AWS services and resources. This implies that any user or entity assigned this role can essentially do anything within the AWS environment, including accessing, modifying, or deleting data in storage systems.

- Role Name: Developer
    - Policies: AmazonEC2FullAccess, AmazonRDSFullAccess, AmazonS3FullAccess
    - The combination of these policies provides broad access to EC2, RDS, and S3.
        - AmazonEC2FullAccess: Grants full access to EC2. This could allow developers to access or modify instances that they shouldn't.
        - AmazonRDSFullAccess: Grants full access to RDS. Developers could potentially access any database, potentially viewing or modifying data.
        - AmazonS3FullAccess: Grants full access to S3. Developers could access, modify, or delete any object in any S3 bucket, including critical data.
- **Resources Potentially Affected**:
    - **Amazon RDS (Relational Database Service)**: The prod-db instance, being a production database, could be accessed, modified, or deleted by anyone with the Admin or Developer role.
    - **Amazon S3 Buckets**: Both the patient records and app-backup buckets could be accessed or modified by anyone with the Admin or Developer role.
    - **EC2 Instances**: Instances such as webserver and appserver could be accessed, modified, or terminated by anyone with the Admin or Developer role.

  The primary issue here is that the permissions granted to both the Admin and Developer roles are exceedingly broad. Such permissions expose a significant risk of unauthorized data access and tampering.

Recommendation:

- **IAM Roles & Policies**: You should refine the IAM roles to adhere to the principle of least privilege. This means only granting permissions that are strictly necessary for the role's responsibilities. For example:
    - Developers might need read access to certain S3 buckets but not write or delete access.
    - Developers might need to launch or terminate specific EC2 instances but not access the production database.
- **Resource-Level Permissions**: AWS also allows for fine-grained access control at the resource level. For instance, we can specify that a particular IAM entity can only access a specific RDS instance or S3 bucket.
- Regularly audit IAM roles and permissions, ensuring they align with job functions.

c. Exposed AWS Access Keys in CloudFormation Outputs

- AWS AccessKeyID and secretAccessKey were found in the CloudFormation Stack output, making them visible and accessible. Secrets hardcoded in CloudFormation outputs pose a significant security risk as they can be exploited by malicious actors or malware, potentially leading to unauthorized access to other AWS services and resources.

Recommendations:

- Immediately remove sensitive access keys and other credentials from CloudFormation outputs. Utilize AWS Identity and Access Management (IAM) roles to grant permissions and access to AWS resources. Avoid hardcoding access keys.
- For sensitive data, consider using AWS Key Management Service (KMS) for encryption and key management. If using access keys, set up a key rotation process to change them regularly for enhanced security.
- Consider using AWS Secrets Manager or Parameter Store for managing secrets securely.

d. Default Database Credentials:

- The RDS instance is configured with default credentials.
  Anyone aware of these credentials can gain unauthorized access, leading to data breaches or database corruption.

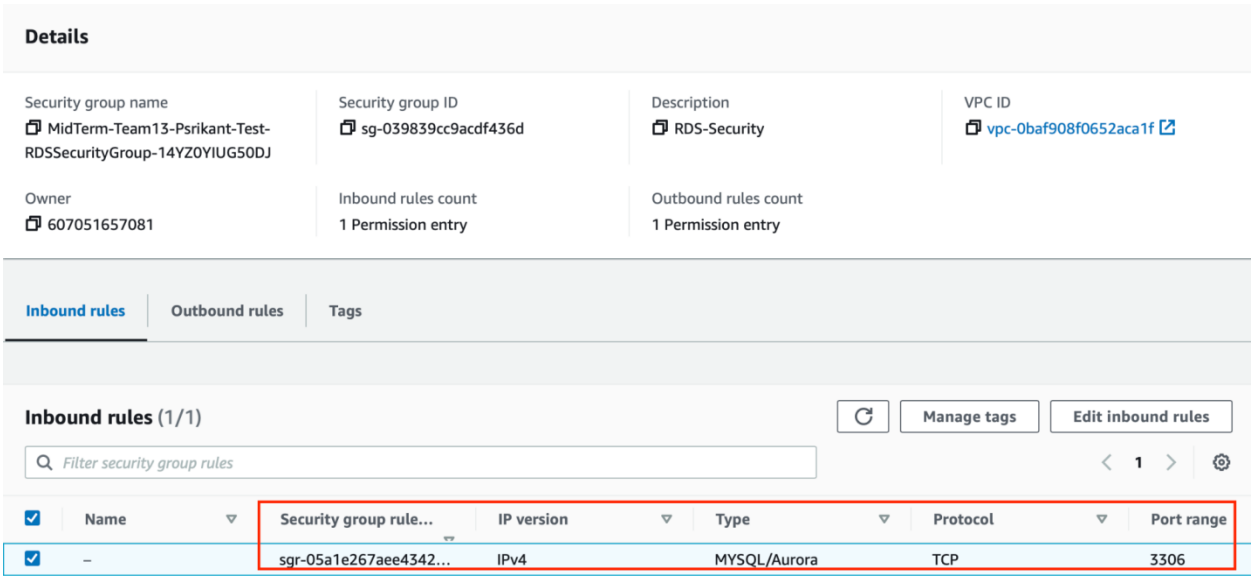
Recommendation:

- Use unique and strong passwords for databases. We should use AWS Secrets Manager for storing and retrieving database credentials.

e. Inadequate Network Security around Data Stores:

Security Groups:

- **Security Group Name**: sg-db
  - Allow all traffic from 0.0.0.0/0 on port 3306 (MySQL)
  - This is a critical security concern. The database port is open to the entire internet. Any malicious entity could potentially try to brute-force or exploit vulnerabilities on this port to gain unauthorized access to the database.



Resources Potentially Affected:

- **Amazon RDS (Relational Database Service)**: The prod-db instance uses the sg-db security group. Given the overly permissive inbound rule of sg-db, this production database is exposed to potential unauthorized access from anywhere on the internet. Tightening these security group rules and ensuring only necessary ports are open to the required IP ranges or security groups would significantly enhance security.

Recommendations:

- Ensure data stores are shielded from unauthorized access.
- Review and refine security groups and firewall rules.
- Isolate data stores in private subnets, ensuring they're not directly accessible from the public internet.

3. Other Recommendations

Regular Data Audits and Monitoring:

- Periodically review and monitor data access patterns.
- Adopt tools like AWS Macie, CloudWatch to detect sensitive data and monitor access.
- Set up real-time alerts for unusual data access patterns.

Data Backup and Disaster Recovery:

- Ensure data integrity and availability.
- Regularly backup data, ensuring backups are encrypted.
- Test backup restoration periodically.
- Develop a comprehensive disaster recovery plan.
- RDS instances should have automatic backups enabled.

Minor Vulnerabilities:

- Deletion protection is not enabled for RDS instances. If deletion protection is there, we can only delete instances that do not have deletion protection enabled.
- RDS instances doesn’t have multi-AZ enabled. In case of failure, with a single-AZ deployment configuration, should an availability zone-specific database failure occur, Amazon RDS cannot automatically fail over to the standby availability zone.
- Macie is not enabled. Using machine learning and pattern matching, Amazon Macie will fully manage data protection and privacy solution that assists us in finding, tracking, and safeguarding sensitive data stored on AWS.
- There isn't a backup vault. An organization's vital data could be lost in the case of an unintentional deletion, system malfunctions, or natural disasters if they don't have an AWS Backup vault.

4. Conclusion

Healthcare Company's data management strategy, while extensive, exhibits several vulnerabilities, especially in encryption and access control. Addressing these vulnerabilities is paramount, not only for the security and trustworthiness of the platform but also for regulatory compliance. A proactive approach towards data security, continuous monitoring, and regular audits will safeguard patient data effectively.

Virtual Machine Vulnerability Assessment Report

Objective: Assess and Address Vulnerabilities in VMs Hosting Healthcare Applications

1. Executive Summary

This report outlines the vulnerabilities discovered in the virtual machine (VM) used by the healthcare organization. The assessment revealed multiple vulnerabilities and misconfigurations associated with the virtual machine.

The EC2 instance, running on Amazon Linux, is operating on an outdated AMI, which could expose it to unpatched vulnerabilities. The security configurations, specifically the open inbound rules on ports 22 and 80 to the entire internet (0.0.0.0/0), increase the risk of unauthorized access attempts. Moreover, monitoring is disabled, which inhibits the ability to detect and respond to malicious activities in real-time.

Furthermore, the instance's termination and stop protections are disabled, posing a risk of unintentional data loss or service disruptions. The storage configuration for the VM is also not optimized for data protection, with the root EBS volume unencrypted. This makes the stored data susceptible to exposure if the volume is ever compromised or accessed. It is crucial to address these vulnerabilities promptly to ensure the integrity, confidentiality, and availability of the services and data hosted on this virtual machine.

2. Identified Vulnerabilities & Potential Impact

a. Instance Vulnerabilities:

- **Monitoring Disabled**:
    - Without monitoring enabled, it's harder to detect performance issues, failures, or suspicious activities on the instance.
    - **Recommendation**: Enable CloudWatch monitoring for the instance. This will allow you to detect performance anomalies, failures, and potentially suspicious activities.

- **AMI Age**:

- The AMI used (amzn2-ami-hvm-2.0.20201218.1-x86_64-gp2) is from December 2020. Given the current date is late 2023, this AMI is almost three years old. It might contain outdated software and known vulnerabilities that have been patched in later releases.
- **Recommendation**: Consider updating the instance to use a more recent AMI. This will ensure you have the latest security patches and software updates.

- **Termination and Stop Protection Disabled**:
  - With termination protection disabled, there's a risk of accidental termination of the instance.
  - Similarly, with stop protection disabled, the instance could be unintentionally stopped.
  - **Recommendation**: Enable termination protection to prevent accidental termination. Consider enabling stop protection if the instance should not be unintentionally stopped.

- **Legacy BIOS Boot Mode**:

  - The instance uses legacy-bios as its boot mode. UEFI is a modern replacement for BIOS and offers better security features. Using legacy BIOS might expose the instance to certain vulnerabilities.
  - **Recommendation**: For new instances, consider using an AMI that supports UEFI boot mode for better security and performance.

b. Networking Vulnerabilities:

- **Publicly Accessible IP:**
  - The instance has a public IPv4 address (52.20.40.112). This means the instance is reachable from the internet, exposing it to potential attacks if not properly secured.
  - **Recommendation**: If the instance doesn't need to be publicly accessible, consider placing it in a private subnet or using a security group to restrict access. If public access is required, ensure that only necessary ports are open and use tools like intrusion detection systems (IDS) to monitor traffic.

- **Security Group Configuration:**

- The security group associated with the instance allows inbound traffic on port 22 (SSH) and port 80 (HTTP) from any source (0.0.0.0/0). This is a major vulnerability.
    - Open SSH (22): Exposes the instance to potential brute-force attacks.
    - Open HTTP (80): If there's a web server running, it's exposed to potential web attacks from anyone on the internet.
- The security group allows all outbound traffic, which, if the instance gets compromised, can be used by an attacker to exfiltrate data or launch further attacks.

- **Recommendation**:
    - Restrict SSH (22) access to specific IPs or IP ranges that require it. Consider using a VPN or AWS's Session Manager for more secure access.
    - If port 80 (HTTP) is not in use, close it. If it's in use, consider using a Web Application Firewall (WAF) like AWS WAF to protect against web attacks.
    - Review outbound rules to restrict unnecessary outbound traffic.

c. Software/Kernal Vulnerabilities:

- **High Vulnerabilities:**
  There are several high vulnerabilities mostly associated with the kernel and some related to other software packages like libxml2-python, p11-kit-trust, and python-urllib3. These vulnerabilities can lead to unauthorized system access, data breaches, denial of service, and other malicious activities.

- **Medium/low Vulnerabilities:**
  The VM has numerous vulnerabilities spread across different software packages. The kernel is especially susceptible, with multiple CVEs (Common Vulnerabilities and Exposures) associated with it. Other software packages like libxml2-python, libstdc++, libgcc, and mariadb-libs also have vulnerabilities that need addressing.

The below vulnerabilities have been obtained by scanning environment using Amazon Inspector 2:

Vulnerability	Critical	High	All
CVE-2023-4921 - kernel	0	1	1
CVE-2023-4641 - shadow-utils	0	0	1
CVE-2023-4623 - kernel	0	1	1
CVE-2023-4622 - kernel	0	1	1
CVE-2023-4459 - kernel	0	0	1
CVE-2023-4387 - kernel	0	0	1
CVE-2023-4208 - kernel	0	1	1
CVE-2023-4207 - kernel	0	1	1
CVE-2023-4206 - kernel	0	1	1
CVE-2023-4128 - kernel	0	0	1
CVE-2023-4039 - libstdc++, libgcc and 1 more	0	0	1
CVE-2023-39615 - libxml2-python, libxml2	0	0	1
CVE-2023-39194 - kernel	0	0	1
CVE-2023-39193 - kernel	0	0	1
CVE-2023-39192 - kernel	0	0	1
CVE-2023-3776 - kernel	0	1	1
CVE-2023-3772 - kernel	0	0	1
CVE-2023-3611 - kernel	0	1	1
CVE-2023-3609 - kernel	0	1	1
CVE-2023-35001 - kernel	0	1	1
CVE-2023-34319 - kernel	0	0	1

CVE-2023-34256 - kernel	0	0	1
CVE-2023-33460 - yajl	0	0	1
CVE-2023-33203 - kernel	0	0	1
CVE-2023-32681 - python-requests	0	0	1
CVE-2023-32233 - kernel	0	1	1
CVE-2023-3212 - kernel	0	0	1
CVE-2023-3161 - kernel	0	0	1
CVE-2023-31436 - kernel	0	0	1
CVE-2023-3117 - kernel	0	1	1
CVE-2023-3111 - kernel	0	0	1
CVE-2023-3090 - kernel	0	1	1
CVE-2023-2985 - kernel	0	0	1
CVE-2023-29469 - libxml2-python, libxml2	0	0	1
CVE-2023-28938 - mdadm	0	0	1
CVE-2023-28772 - kernel	0	1	1
CVE-2023-28736 - mdadm	0	0	1
CVE-2023-2860 - kernel	0	0	1
CVE-2023-28484 - libxml2-python, libxml2	0	0	1
CVE-2023-28466 - kernel	0	1	1
CVE-2023-26545 - kernel	0	1	1
CVE-2023-26112 - python-configobj	0	0	1
CVE-2023-2602 - libcap	0	0	1

CVE-2023-2513 - kernel	0	0	1
CVE-2023-23455 - kernel	0	0	1
CVE-2023-23454 - kernel	0	0	1
CVE-2023-2269 - kernel	0	0	1
CVE-2023-2194 - kernel	0	0	1
CVE-2023-2162 - kernel	0	0	1
CVE-2023-2124 - kernel	0	1	1
CVE-2023-20588 - kernel	0	0	1
CVE-2023-20569 - kernel	0	0	1
CVE-2023-2002 - kernel	0	0	1
CVE-2023-1998 - kernel	0	0	1
CVE-2023-1838 - kernel	0	1	1
CVE-2023-1829 - kernel	0	1	1
CVE-2023-1637 - kernel	0	0	1
CVE-2023-1281 - kernel	0	1	1
CVE-2023-1206 - kernel	0	1	1
CVE-2023-0459 - kernel	0	1	1
CVE-2023-0458 - kernel	0	1	1
CVE-2023-0394 - kernel	0	0	1
CVE-2023-0045 - kernel	0	1	1
CVE-2022-47929 - kernel	0	0	1
CVE-2022-45934 - kernel	0	1	1

CVE-2022-43750 - kernel	0	1	1
CVE-2022-42896 - kernel	0	0	1
CVE-2022-42895 - kernel	0	0	1
CVE-2022-42329 - kernel	0	0	1
CVE-2022-42328 - kernel	0	0	1
CVE-2022-41850 - kernel	0	0	1
CVE-2022-41849 - kernel	0	0	1
CVE-2022-40897 - python2-setuptools	0	0	1
CVE-2022-40768 - kernel	0	0	1
CVE-2022-40307 - kernel	0	0	1
CVE-2022-40304 - libxml2-python, libxml2	0	1	1
CVE-2022-40303 - libxml2-python, libxml2	0	1	1
CVE-2022-39842 - kernel	0	0	1
CVE-2022-39188 - kernel	0	0	1
CVE-2022-36946 - kernel	0	0	1
CVE-2022-36879 - kernel	0	0	1
CVE-2022-3649 - kernel	0	1	1
CVE-2022-3646 - kernel	0	0	1
CVE-2022-3643 - kernel	0	0	1
CVE-2022-3621 - kernel	0	1	1
CVE-2022-36123 - kernel	0	1	1
CVE-2022-3594 - kernel	0	1	1

CVE-2022-3565 - kernel	0	1	1
CVE-2022-3564 - kernel	0	1	1
CVE-2022-3542 - kernel	0	0	1
CVE-2022-3524 - kernel	0	1	1
CVE-2022-34918 - kernel	0	1	1
CVE-2022-34903 - gnupg2	0	0	1
CVE-2022-33744 - kernel	0	0	1
CVE-2022-33742 - kernel	0	1	1
CVE-2022-33741 - kernel	0	1	1
CVE-2022-33740 - kernel	0	1	1
CVE-2022-32981 - kernel	0	1	1
CVE-2022-32296 - kernel	0	0	1
CVE-2022-32250 - kernel	0	1	1
CVE-2022-31624 - mariadb-libs	0	0	1
CVE-2022-30594 - kernel	0	1	1
CVE-2022-3028 - kernel	0	0	1
CVE-2022-29901 - kernel	0	0	1
CVE-2022-29824 - libxml2-python, libxml2	0	1	1
CVE-2022-2978 - kernel	0	1	1
CVE-2022-29581 - kernel	0	1	1
CVE-2022-28693 - kernel	0	0	1
CVE-2022-28390 - kernel	0	1	1

CVE-2022-28389 - kernel	0	1	1
CVE-2022-28356 - kernel	0	0	1
CVE-2022-27666 - kernel	0	1	1
CVE-2022-27406 - freetype	0	1	1
CVE-2022-27405 - freetype	0	1	1
CVE-2022-27404 - freetype	0	1	1
CVE-2022-27384 - mariadb-libs	0	1	1
CVE-2022-27380 - mariadb-libs	0	1	1
CVE-2022-2663 - kernel	0	0	1
CVE-2022-26490 - kernel	0	1	1
CVE-2022-2639 - kernel	0	1	1
CVE-2022-26373 - kernel	0	0	1
CVE-2022-26365 - kernel	0	1	1
CVE-2022-2588 - kernel	0	1	1
CVE-2022-2586 - kernel	0	0	1
CVE-2022-24795 - yajl	0	0	1
CVE-2022-24448 - kernel	0	0	1
CVE-2022-23960 - kernel	0	0	1
CVE-2022-23308 - libxml2-python, libxml2	0	1	1
CVE-2022-2318 - kernel	0	0	1
CVE-2022-2153 - kernel	0	0	1
CVE-2022-21166 - kernel	0	0	1

CVE-2022-21125 - kernel	0	0	1
CVE-2022-21123 - kernel	0	0	1
CVE-2022-20369 - kernel	0	0	1
CVE-2022-20368 - kernel	0	1	1
CVE-2022-20141 - kernel	0	1	1
CVE-2022-1966 - kernel	0	1	1
CVE-2022-1729 - kernel	0	1	1
CVE-2022-1679 - kernel	0	1	1
CVE-2022-1586 - pcre2	0	1	1
CVE-2022-1516 - kernel	0	0	1
CVE-2022-1462 - kernel	0	0	1
CVE-2022-1353 - kernel	0	0	1
CVE-2022-1304 - e2fsprogs, libss and 2 more	0	0	1
CVE-2022-1210 - jbigkit-libs	0	0	1
CVE-2022-1184 - kernel	0	0	1
CVE-2022-1016 - kernel	0	0	1
CVE-2022-1015 - kernel	0	0	1
CVE-2022-1012 - kernel	0	1	1
CVE-2022-1011 - kernel	0	1	1
CVE-2022-0854 - kernel	0	0	1
CVE-2022-0812 - kernel	0	0	1
CVE-2022-0617 - kernel	0	0	1

CVE-2022-0563 - libmount, libblkid and 4 more	0	0	1
CVE-2022-0494 - kernel	0	0	1
CVE-2022-0492 - kernel	0	1	1
CVE-2022-0435 - kernel	0	1	1
CVE-2022-0330 - kernel	0	1	1
CVE-2022-0002 - kernel	0	0	1
CVE-2022-0001 - kernel	0	0	1
CVE-2021-46668 - mariadb-libs	0	0	1
CVE-2021-46667 - mariadb-libs	0	0	1
CVE-2021-46666 - mariadb-libs	0	0	1
CVE-2021-46663 - mariadb-libs	0	0	1
CVE-2021-46661 - mariadb-libs	0	0	1
CVE-2021-46659 - mariadb-libs	0	0	1
CVE-2021-46657 - mariadb-libs	0	0	1
CVE-2021-44733 - kernel	0	1	1
CVE-2021-42771 - python-babel	0	1	1
CVE-2021-42574 - libstdc++, libgcc and 1 more	0	1	1
CVE-2021-4197 - kernel	0	0	1
CVE-2021-41864 - kernel	0	1	1
CVE-2021-4159 - kernel	0	0	1
CVE-2021-4155 - kernel	0	0	1
CVE-2021-4083 - kernel	0	1	1

CVE-2021-40490 - kernel	0	1	1
CVE-2021-4002 - kernel	0	0	1
CVE-2021-39648 - kernel	0	0	1
CVE-2021-3923 - kernel	0	0	1
CVE-2021-38300 - kernel	0	1	1
CVE-2021-38199 - kernel	0	0	1
CVE-2021-38198 - kernel	0	0	1
CVE-2021-38185 - cpio	0	1	1
CVE-2021-3772 - kernel	0	0	1
CVE-2021-3764 - kernel	0	0	1
CVE-2021-3753 - kernel	0	0	1
CVE-2021-3744 - kernel	0	0	1
CVE-2021-3732 - kernel	0	0	1
CVE-2021-37159 - kernel	0	0	1
CVE-2021-3656 - kernel	0	1	1
CVE-2021-3655 - kernel	0	0	1
CVE-2021-3653 - kernel	0	1	1
CVE-2021-3640 - kernel	0	0	1
CVE-2021-36087 - libsepol	0	0	1
CVE-2021-36086 - libsepol	0	0	1
CVE-2021-36085 - libsepol	0	0	1
CVE-2021-36084 - libsepol	0	0	1

CVE-2021-3573 - kernel	0	0	1
CVE-2021-3564 - kernel	0	0	1
CVE-2021-35477 - kernel	0	0	1
CVE-2021-3541 - libxml2-python, libxml2	0	0	1
CVE-2021-3537 - libxml2-python, libxml2	0	1	1
CVE-2021-3518 - libxml2-python, libxml2	0	1	1
CVE-2021-3517 - libxml2-python, libxml2	0	1	1
CVE-2021-3516 - libxml2-python, libxml2	0	1	1
CVE-2021-3483 - kernel	0	1	1
CVE-2021-34556 - kernel	0	0	1
CVE-2021-3429 - cloud-init	0	0	1
CVE-2021-3421 - rpm-plugin-systemd-inhibit, rpm-build-libs and 3 more	0	0	1
CVE-2021-33909 - kernel	0	1	1
CVE-2021-33655 - kernel	0	0	1
CVE-2021-33624 - kernel	0	0	1
CVE-2021-33560 - libgcrypt	0	1	1
CVE-2021-33503 - python-urllib3	0	1	1
CVE-2021-3348 - kernel	0	1	1
CVE-2021-3347 - kernel	0	1	1
CVE-2021-33294 - elfutils-libs, elfutils-libelf and 1 more	0	0	1
CVE-2021-33200 - kernel	0	1	1
CVE-2021-33034 - kernel	0	1	1

CVE-2021-32399 - kernel	0	1	1
CVE-2021-31829 - kernel	0	0	1
CVE-2021-3178 - kernel	0	0	1
CVE-2021-29650 - kernel	0	0	1
CVE-2021-29647 - kernel	0	0	1
CVE-2021-29265 - kernel	0	0	1
CVE-2021-29155 - kernel	0	0	1
CVE-2021-29154 - kernel	0	1	1
CVE-2021-28972 - kernel	0	0	1
CVE-2021-28964 - kernel	0	0	1
CVE-2021-28715 - kernel	0	0	1
CVE-2021-28714 - kernel	0	0	1
CVE-2021-28713 - kernel	0	0	1
CVE-2021-28712 - kernel	0	0	1
CVE-2021-28711 - kernel	0	0	1
CVE-2021-28688 - kernel	0	0	1
CVE-2021-28660 - kernel	0	1	1
CVE-2021-28038 - kernel	0	0	1
CVE-2021-27365 - kernel	0	1	1
CVE-2021-27364 - kernel	0	1	1
CVE-2021-27363 - kernel	0	0	1
CVE-2021-26932 - kernel	0	0	1

CVE-2021-26931 - kernel	0	0	1
CVE-2021-26930 - kernel	0	1	1
CVE-2021-26401 - kernel	0	0	1
CVE-2021-26341 - kernel	0	0	1
CVE-2021-23133 - kernel	0	1	1
CVE-2021-22555 - kernel	0	1	1
CVE-2021-22543 - kernel	0	1	1
CVE-2021-20321 - kernel	0	0	1
CVE-2021-20317 - kernel	0	0	1
CVE-2021-20271 - rpm-plugin-systemd-inhibit, rpm-build-libs and 3 more	0	0	1
CVE-2021-0129 - kernel	0	0	1
CVE-2020-8632 - cloud-init	0	1	1
CVE-2020-8631 - cloud-init	0	1	1
CVE-2020-36322 - kernel	0	0	1
CVE-2020-29661 - kernel	0	1	1
CVE-2020-29660 - kernel	0	0	1
CVE-2020-29569 - kernel	0	1	1
CVE-2020-29568 - kernel	0	0	1
CVE-2020-29374 - kernel	0	0	1
CVE-2020-29363 - p11-kit-trust, p11-kit	0	1	1
CVE-2020-29362 - p11-kit-trust, p11-kit	0	0	1
CVE-2020-29361 - p11-kit-trust, p11-kit	0	1	1

CVE-2020-28374 - kernel	0	1	1
CVE-2020-27825 - kernel	0	0	1
CVE-2020-27815 - kernel	0	1	1
CVE-2020-27171 - kernel	0	0	1
CVE-2020-26558 - kernel	0	0	1
CVE-2020-26137 - python-urllib3	0	0	1
CVE-2020-25672 - kernel	0	1	1
CVE-2020-25671 - kernel	0	1	1
CVE-2020-25670 - kernel	0	1	1
CVE-2020-25658 - python2-rsa	0	0	1
CVE-2020-24977 - libxml2-python, libxml2	0	0	1
CVE-2020-21913 - libicu	0	0	1
CVE-2020-21047 - elfutils-libs, elfutils-libelf and 1 more	0	0	1
CVE-2020-16119 - kernel	0	1	1
CVE-2020-14422 - python-ipaddress	0	0	1
CVE-2020-14367 - chrony	0	0	1
CVE-2020-12762 - libfastjson	0	1	1
CVE-2019-7308 - kernel	0	0	1
CVE-2019-19816 - kernel	0	1	1
CVE-2019-19813 - kernel	0	0	1
CVE-2019-19060 - kernel	0	1	1
CVE-2019-15167 - tcpdump	0	0	1

CVE-2019-12900 - bzip2-libs, bzip2	0	0	1
CVE-2018-7738 - libmount, libblkid and 4 more	0	0	1
CVE-2018-25020 - kernel	0	1	1
CVE-2018-10896 - cloud-init	0	0	1
CVE-2015-8394 - pcre	0	0	1
CVE-2015-8390 - pcre	0	0	1
ALAS2-2023-1920 - libmount, libblkid and 4 more	0	0	1
ALAS2-2021-1634 - kernel	0	0	1

- **Recommendations:**
  - **Kernel Patching:** Given the number of vulnerabilities associated with the kernel, it is recommended to update the kernel to the latest stable version. Ensure to test the update in a staging environment before deploying it in production.
  - **Software Patching:** Regularly update all software packages. Given the number of vulnerabilities, prioritize those with high vulnerabilities first.

d. Storage Details Vulnerabilities:

- **Unencrypted Root Volume**:
  - The root volume (vol-0b8ea28aed082bb7b) is not encrypted. This poses a risk as someone with physical access to the underlying hardware (unlikely in a cloud environment, but possible) could potentially read the data. Additionally, if a snapshot of this volume is taken, it would also be unencrypted.
  - **Recommendation**: For enhanced data protection, consider using encrypted EBS volumes. If encrypting the existing volume is challenging, you can create a snapshot, copy it with encryption, and then create a new volume from the encrypted snapshot.

Minor Vulnerabilities:

- EC2 Instance i-09283702134c21f16 has IMDSv2 disabled or not required. Using IMDSv2 will protect from misconfiguration and SSRF vulnerabilities. IMDSv1 will not.
- EC2 Instance i-09283702134c21f16 not associated with an Instance Profile Role. AWS access from within AWS instances can be done by either encoding AWS keys into AWS API calls or by assigning the instance to a role which has an appropriate permissions policy for the required access. AWS IAM roles reduce the risks associated with sharing and rotating credentials that can be used outside of AWS itself. If credentials are compromised, they can be used from outside of the AWS account.

Conclusion:

Upon completion of our in-depth vulnerability assessment of the hospital's virtual machine infrastructure, maintaining a robust and secure digital environment is paramount in the healthcare sector. The findings highlight several critical areas, including outdated software, unencrypted patient data, and certain security misconfigurations, which need urgent attention to ensure the confidentiality, integrity, and availability of patient data and medical services. It's imperative to understand that in a healthcare setting, these vulnerabilities not only pose risks of regulatory non-compliance, such as with HIPAA, but also have the potential to compromise patient trust and the hospital's reputation. As healthcare continues its rapid digital transformation, we strongly recommend the hospital to adopt a proactive approach by scheduling regular assessments and updates, ensuring that its virtual infrastructure is always a step ahead in terms of security and resilience.

Network Security Assessment Report

Objective: Assess Network Vulnerabilities and Propose Secure Access Control Measures

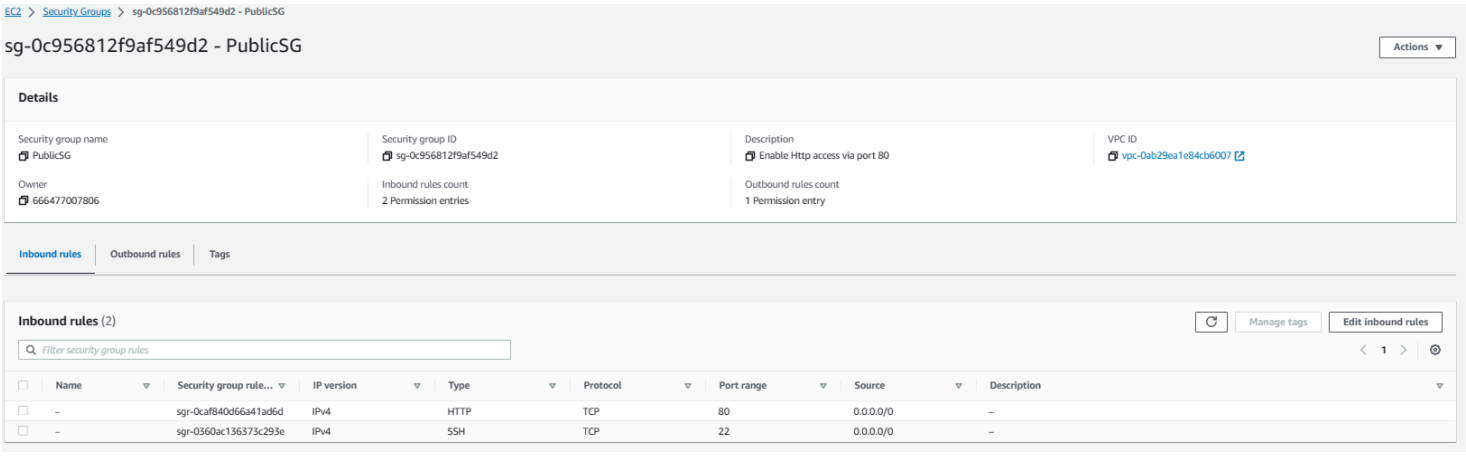
1.Executive Summary

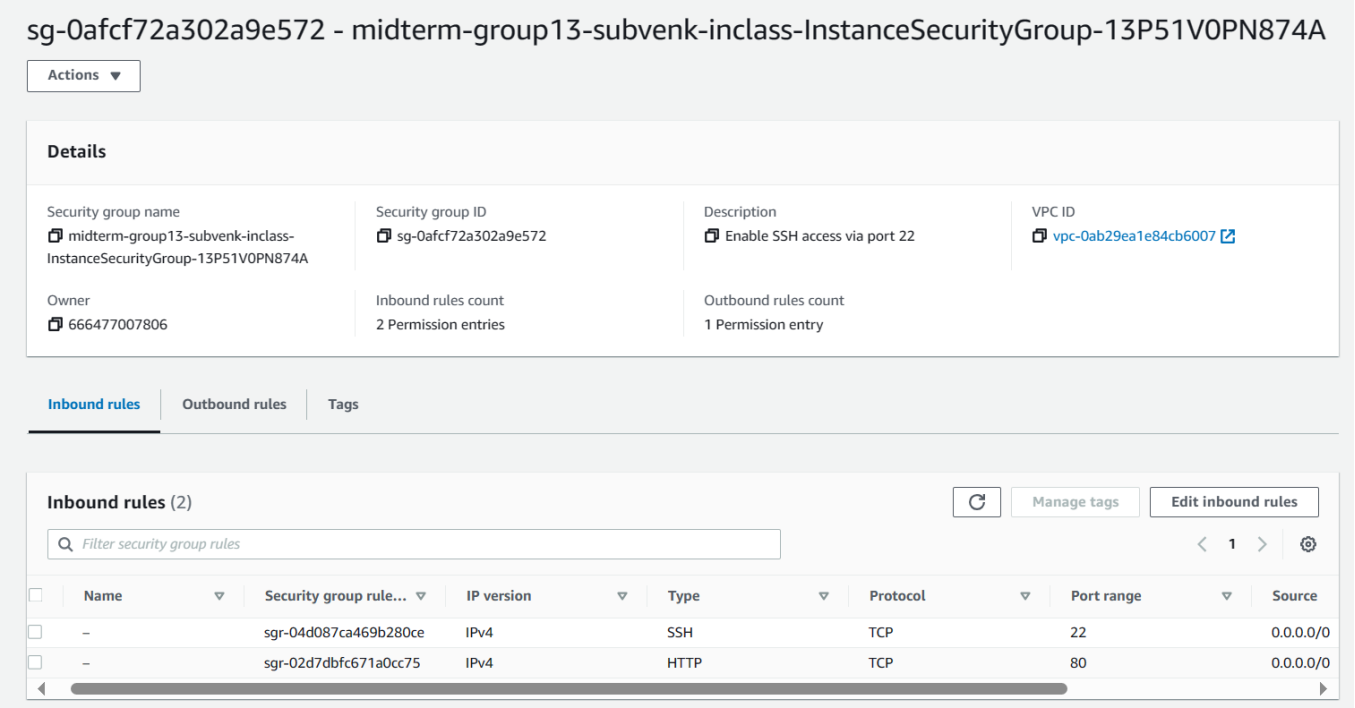
Network security in an AWS environment is crucial because it safeguards sensitive data, ensures service continuity, and meets industry-specific compliance standards. Given AWS's shared responsibility model, while AWS secures the underlying infrastructure, users must protect their data and applications. Effective network security not only upholds a company's reputation but also prevents substantial financial losses stemming from potential breaches or non-compliance penalties.

2. Identified Risks & Potential Impact

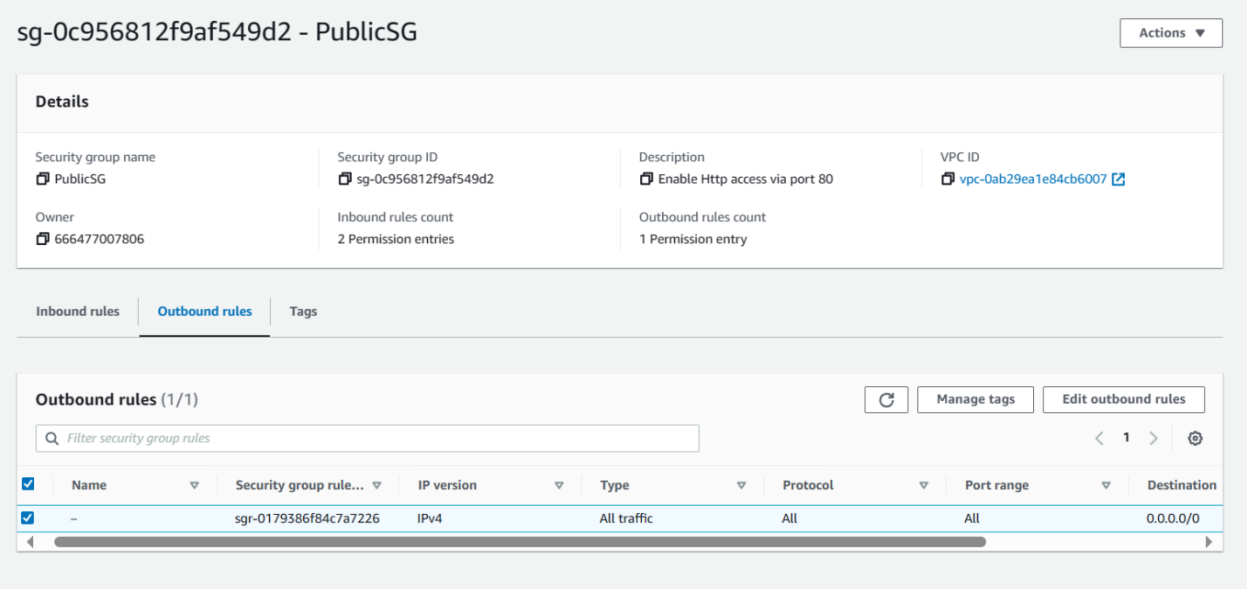
a. Open Network Ports and Misconfigured Security Groups:

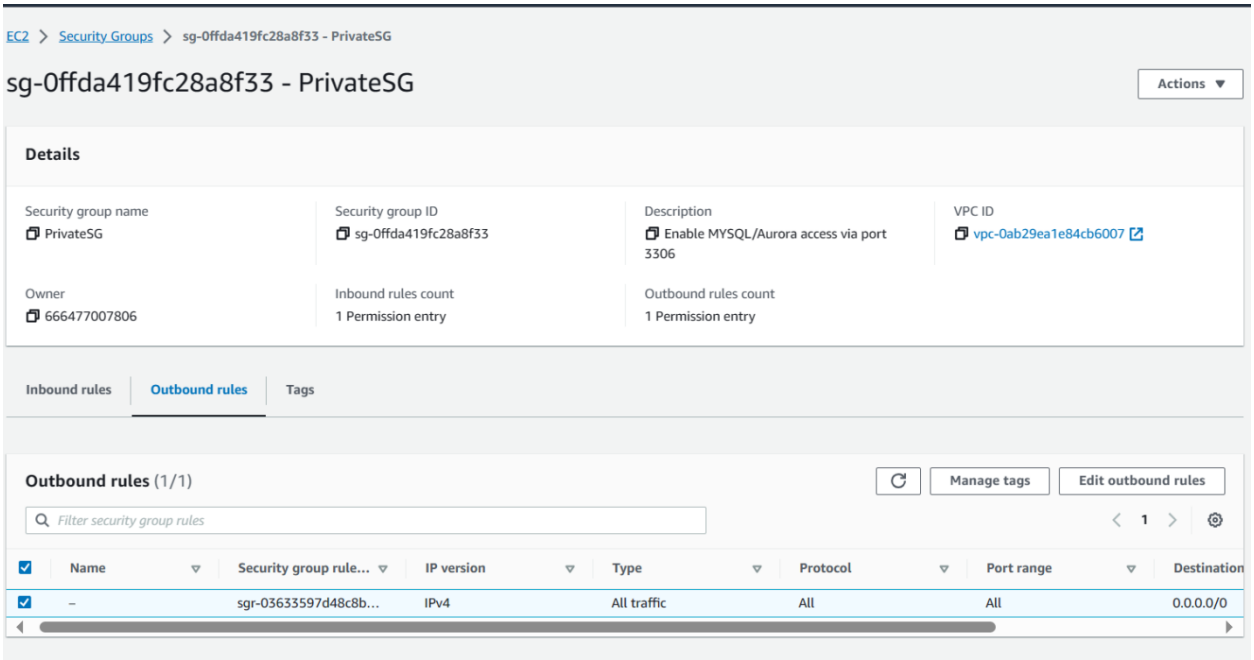
- In our scenario, for the security group PublicSG and InstanceSecurityGroup, if we see the inbound rules, we can see that ingress from 0.0.0.0/0 (i.e the entire internet) to SSH port 22 is allowed. Leaving port 22 open can expose resources to potential unauthorized access attempts and brute force attacks. Where the attacker runs a script to crack the password for the server by trying all possible password combinations.

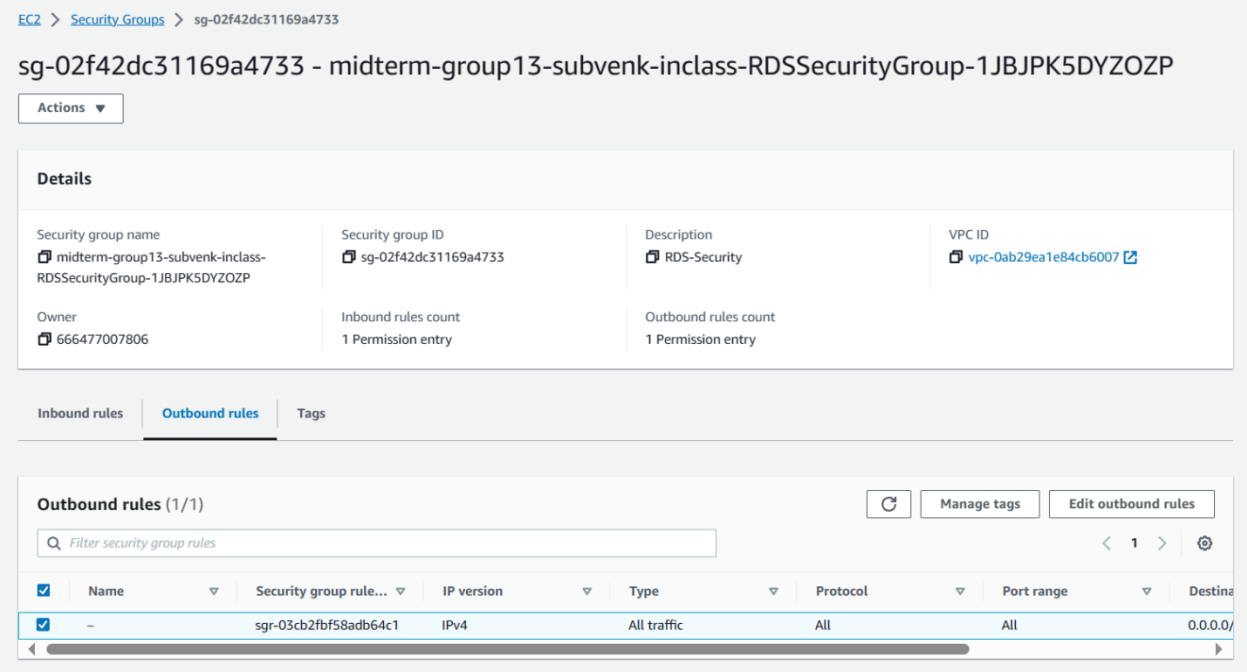


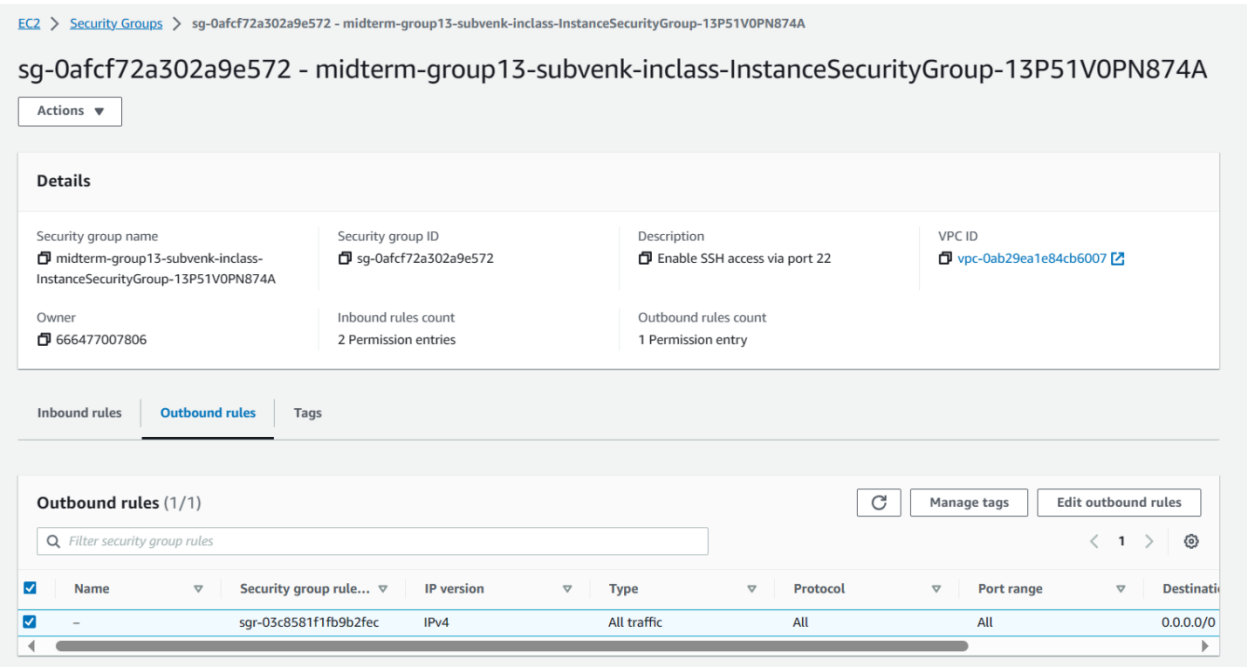


- **Recommendation**:
    - As a best practice, we can turn off port 22 ingress, or at least limit the ingress rule to specific IP addresses or IP ranges that need SSH access. This reduces the attack surface. We should use a Zero Trust approach and narrow ingress traffic as much as possible.
    - Users can also consider accessing resources through a VPN or AWS Direct Connect rather than directly over the internet.
- We can also see that port 80 ingress is enabled for the PublicSG; this is a common practice while hosting a publicly accessible web application. The advantage is that the application or website can be accessed by users using any IP address. The disadvantage is that this public port can become a target for various types of attacks, like a DDoS attack or attempts to exploit potential vulnerabilities in the hosted application.
- **Recommendations:**
    - To mitigate this, we can consider using AWS (Web Application Firewall) WAF to help and protect our application from common web threats by creating web ACLs (Access Control Lists).
    - We can also enable AWS CloudWatch and AWS CloudTrail for detecting and responding to suspicious activities.


sg-0c956812f9af549d2 - PublicSG
Actions
Details
Security group name
PublicSG
Security group ID
sg-0c956812f9af549d2
Description
Enable Http access via port 80
VPC ID
vpc-0ab29ea1e84cb6007
Owner
666477007806
Inbound rules count
2 Permission entries
Outbound rules count
1 Permission entry
Inbound rules
Outbound rules
Tags
Outbound rules (1/1)
Manage tags
Edit outbound rules
Filter security group rules
Name
Security group rule...
IP version
Type
Protocol
Port range
Destination
–
sgr-0179386f84c7a7226
IPv4
All traffic
All
All
0.0.0.0/0


EC2
Security Groups
sg-0ffda419fc28a8f33 - PrivateSG
sg-0ffda419fc28a8f33 - PrivateSG
Actions
Details
Security group name
PrivateSG
Security group ID
sg-0ffda419fc28a8f33
Description
Enable MYSQL/Aurora access via port 3306
VPC ID
vpc-0ab29ea1e84cb6007
Owner
666477007806
Inbound rules count
1 Permission entry
Outbound rules count
1 Permission entry
Inbound rules
Outbound rules
Tags
Outbound rules (1/1)
Manage tags
Edit outbound rules
Filter security group rules
Name
Security group rule...
IP version
Type
Protocol
Port range
Destination
–
sgr-03633597d48c8b...
IPv4
All traffic
All
All
0.0.0.0/0



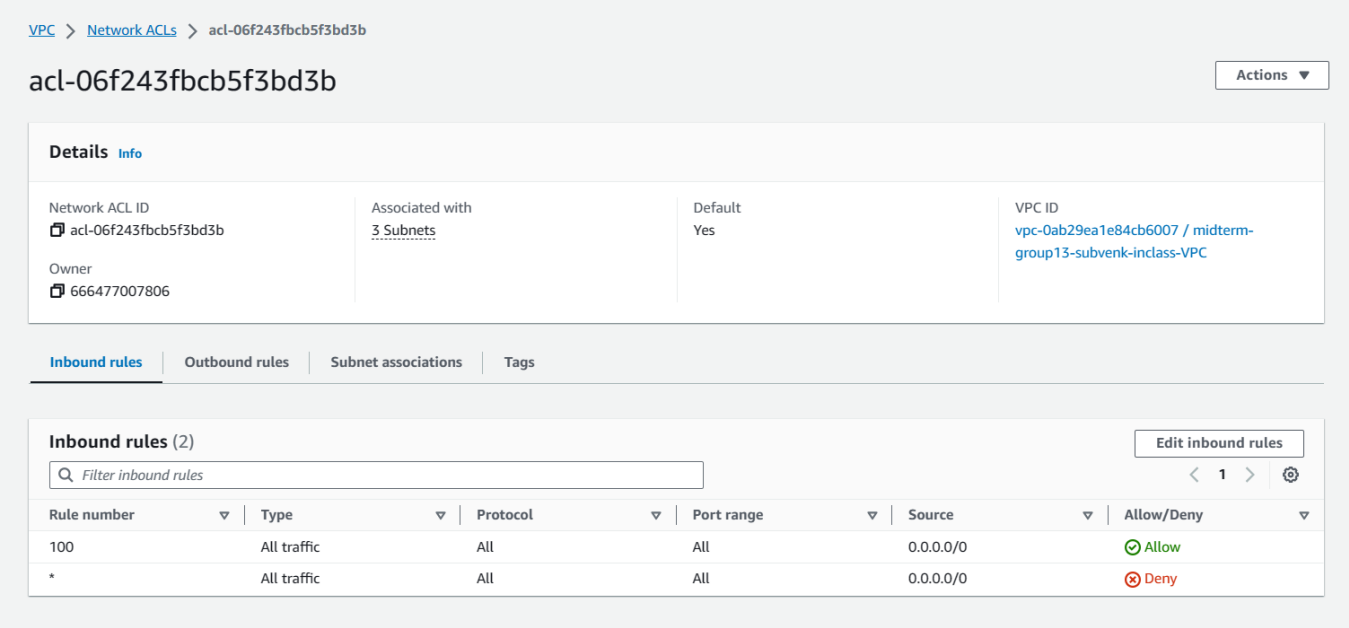


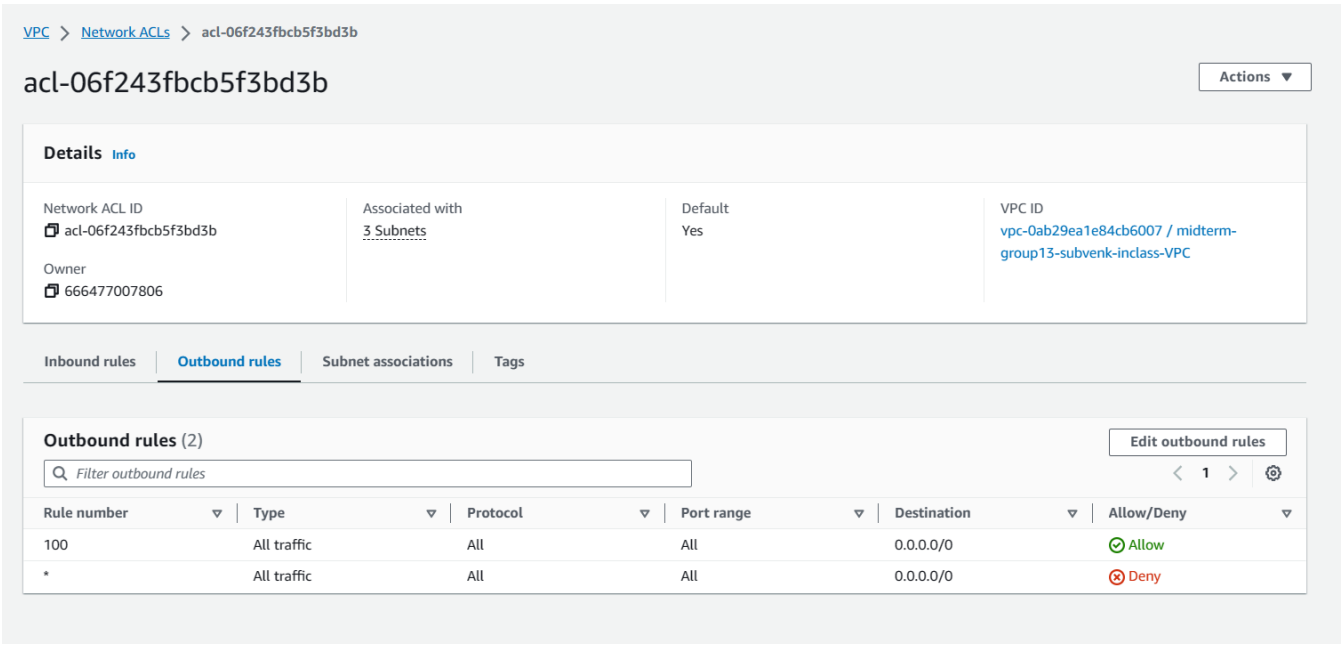
- As shown in the above screenshots, all the security groups have egress/outbound traffic permission to the entire internet (0.0.0.0/0).

- **Recommendations**:
  - We should limit outbound traffic to specific IP addresses based on the needs of the application.

- Unlimited access to the internet should only be provided only if a particular application requires it. For the rest of the resources, we should limit outbound traffic, so that in case of a breach, data can't be exfiltrated quickly.
- Limited outbound access also helps in preventing data leaks.

b. Network ACLs with overly permissive configurations:





- When we check the ingress and egress rules of the Network ACL(NACL), we can see that this NACL has every port open to the internet as a part of rule 100.

- This NACL is applied to a subnet which has our EC2 instance for the medical company. Since this NACL allows ingress from the entire internet, it means that the NACL is not acting as a barrier to any specific type of inbound traffic.

- **The associated risks are:**

    - Increased Attack Surface – The EC2 instance is potentially exposed to any form of traffic from the internet. Instance may not be directly accessible, but it does mean that traffic can reach the subnet level.
    - Potential Unauthorized Access – If the EC2 instance has security groups or application-level configurations that are also overly permissive, it can be accessed or attacked from any location.
    - Amplified Impact of Misconfigurations: If there's a misconfiguration at the security group or application level, the permissive NACL amplifies its impact by not providing an additional layer of protection.
    - Denial-of-Service (DoS) Attacks: The NACL won't prevent any form of traffic, making the subnet's resources potential targets for DoS attacks.
    - Data Exfiltration: Egress is also allowed universally, so a compromised instance can send data out to any destination.
    - Bypassing Best Practices: AWS recommends using NACLs as an additional layer of security to blacklist or whitelist specific types of traffic, complementing the security group rules. A permissive rule negates this recommendation.

- **Recommendations**:

    - Security Groups: We need to ensure that the security groups associated with the EC2 instance are restrictive, only allowing necessary traffic.

    - Host-level Firewalls: Consider using host-level firewalls within the EC2 instance for an additional layer of protection.

    - Monitoring and Logging: Implement AWS CloudWatch and VPC Flow Logs to monitor traffic and detect suspicious activities.

    - Principle of Least Privilege: We need to update our NACL rules to only allow traffic that is explicitly required for our medical applications and services to function properly. All other traffic should be denied by default.

    - Explicit Deny: Although NACLs automatically have an implicit deny rule at the end, explicit deny rules can be added for known malicious IP addresses or ranges.

    - Block Unnecessary Ports: Only open the ports that are necessary. For example, if you don’t need SSH or RDP access, ensure those ports are not allowed in your NACL.

- Restrict Egress Traffic: Just as with ingress, we need to apply the principle of least privilege to egress (outbound) traffic. Outbound traffic should be limited to known, necessary destinations.

- Backup and Versioning: Before making changes, we should backup our current NACL configurations as AWS doesn't provide versioning for NACLs. So, we need to manually document changes or use infrastructure as code practices (e.g., CloudFormation, Terraform) to maintain versions of our configurations.

- Separate NACLs for Different Subnets: If you have multiple subnets serving different purposes (e.g., web servers vs. database servers), use separate NACLs for each type of subnet. This way, you can customize rules based on the specific needs and risks of each subnet.

c. Lack of Network Segmentation:

- The network might not have clear demarcations or segmentation for different services, leading to a flat network structure. Breach in one segment could compromise the entire network, leading to lateral movement of attackers.

- **Recommendations:** Segment the network based on departments, services, or data sensitivity levels.
  Steps:
  - Use Virtual Private Clouds (VPCs) or VLANs for segmentation.
  - Ensure strict access controls between segments.
  - Regularly review and update segmentation rules.

d. Inadequate Intrusion Detection and Prevention:

- Absence or inadequate setup of Intrusion Detection and Prevention Systems (IDPS). Undetected malicious activities, unauthorized access, and potential data breaches.
- **Recommendations:** Monitor and prevent unauthorized network activities.
  Steps:
  - Use solutions like AWS GuardDuty or Cisco's IDS/IPS systems.
  - Regularly update the system with the latest threat signatures.
  - Set up alerts for suspicious activities.

Minor Vulnerabilities:

- VPC vpc-041e909fcefefe34c Flow logs are disabled. For all VPCs in our medical company's cloud infrastructure, Flow logs have been disabled. VPC Flow Logs offer

insight into the network traffic within a VPC, enabling the detection of unusual activity and aiding in security-related tasks.

- By default, VPC subnet subnet-0e84af748d6d90488 automatically allocates public IP addresses. Each VPC subnet operates under its distinct set of traffic rules. However, the automatic assignment of public IPs during launch can unintentionally expose instances within this subnet to the internet. It is advisable to review and adjust this setting to 'No' after creating the subnet to enhance security.

4. Conclusion

It's always best to follow the principle of least privilege and only allow traffic that is explicitly needed. Network security is the bedrock of any organization's cybersecurity strategy, especially for sectors dealing with sensitive data like healthcare. Healthcare Company's current infrastructure presents multiple avenues for potential breaches. However, with proactive measures, stringent access control, and continuous monitoring, the network can be fortified against both present and emerging threats.

Disaster Recovery Assessment Report

Objective: Evaluate Risks Associated with Data Loss and Downtime; Propose a Comprehensive Backup and Disaster Recovery Plan

1. Executive Summary

Healthcare Company's reliance on cloud infrastructure for managing sensitive patient data underscores the need for a robust disaster recovery (DR) strategy. This report identifies the current risks associated with potential data loss and extended downtime and proposes a comprehensive plan to ensure data integrity and availability during unforeseen disasters.

2. Identified Risks & Potential Impact

a. Lack of Redundant Data Storage:

- Data stored in a singular location without redundancy. A failure in the primary storage could result in irreversible data loss.

- **Recommendation:** Ensure data is regularly backed up to secure locations.
    - Implement automated backups using services like AWS Backup or Azure Backup.
    - Store backups in different storage tiers based on data access needs, utilizing services like Amazon S3 Glacier for infrequently accessed backup data.

    - Use AWS Backup to create backup vaults for your critical data and services.

b. No Defined Recovery Point Objective (RPO) and Recovery Time Objective (RTO):

- Absence of clear benchmarks for data recovery and system restoration. Extended downtime, loss of recent data, and inability to meet service level agreements.

- **Recommendation:** Establish clear objectives for data recovery and system restoration.
    - Determine the acceptable data loss window (RPO) and the maximum acceptable downtime (RTO) for each service.
    - Design the DR strategy to meet or exceed these benchmarks.

c. Absence of Regular Backup Testing:

- Backups, if present, aren't regularly tested for integrity and viability. Backups might be corrupted, outdated, or incomplete, making recovery efforts futile.

- **Recommendation:** Ensure that backups are viable for recovery.
    - Periodically restore backups to test environments to verify data integrity.

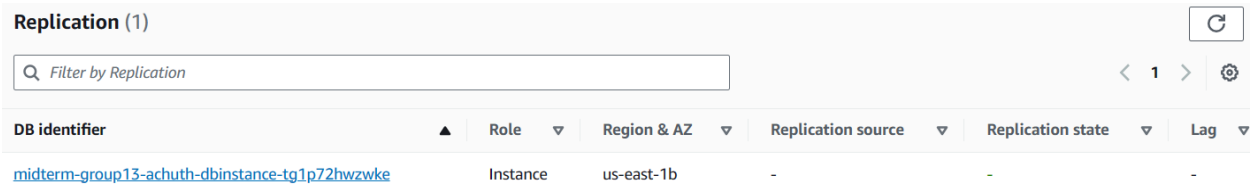
- Review and update backup procedures based on test outcomes.

d. No Geographic Redundancy:

- All resources and backups are in a single geographic region. Regional disasters, like natural calamities or data center outages, could disrupt services and cause data loss.

- **Recommendation:** Ensure that data and services are resilient to regional disasters.
  - Replicate critical data and services across multiple geographic regions.
  - Consider using services like Amazon RDS Global Databases or Azure Geo-Redundant Storage.

  - RDS for the healthcare company is replicated in the same region. However, replicating data across multiple geographic regions is recommended.



e. Lack of Multi-tiered DR Strategy:

- There is no multi-tiered DR strategy implemented now which caters to different disaster levels.

- **Recommendation:** Implement a DR strategy that caters to different disaster levels.
  - **Tier 1:** Localized failures (e.g., VM crash). Solution: Local backups and auto-scaling.
  - **Tier 2:** Data center outages. Solution: Inter-data center replication and failover.
  - **Tier 3:** Regional disasters. Solution: Cross-region replication and failover.

4. Conclusion

For Healthcare Company, ensuring the availability and integrity of patient data is paramount. The current lack of a comprehensive DR plan exposes the organization to significant risks. By implementing the recommended measures, Healthcare Company can safeguard against data loss, minimize downtime, and ensure continued trust from its patients and stakeholders.

References

Amazon Timestream - Developer Guide, docs.aws.amazon.com/timestream/latest/developerguide/pdfs/timestream/latest/developerguide/timestream.pdf. Accessed 27 Oct. 2023.

Amazon Virtual Private Cloud - Docs.Aws.Amazon.Com, docs.aws.amazon.com/pdfs/vpc/latest/userguide/vpc-ug.pdf. Accessed 24 Oct. 2023.

Applying Security Practices to a Network Workload on AWS For ..., docs.aws.amazon.com/pdfs/whitepapers/latest/applying-security-practices-to-network-workload-for-csps/applying-security-practices-to-network-workload-for-csps.pdf. Accessed 27 Oct. 2023.

AWS WAF, AWS Firewall Manager, and AWS Shield Advanced - Developer Guide, docs.aws.amazon.com/waf/latest/developerguide/waf-dg.pdf. Accessed 24 Oct. 2023.

Welcome to AWS Documentation - Docs.Aws.Amazon.Com, docs.aws.amazon.com/. Accessed 23 Oct. 2023.

Welcome to AWS Documentation, docs.aws.amazon.com/pdfs/IAM/latest/UserGuide/iam-ug.pdf. Accessed 24 Oct. 2023.