AWS

VAPT Report

Table of Contents

Document Detail

Document:	AWS Pentest Report
Report Content:	Summarized Technical Report
Classification:	Confidential
Status:	Initial Report
Version:	1.0
Date:	4th Feb 2025
Approved By:	Mr. Aaryan Saharan

Report History

ISSUE DATE	VERSION	DESCRIPTION	PREPARAED BY	
4th Feb 2025	1.0	Initial Report	Mr. Aaryan Saharan	

Executive Summary:

The purpose of this Vulnerability Assessment and Penetration Testing (VAPT) report is to present the findings and recommendations from a penetration test conducted on the AWS environment. The objective was to identify security weaknesses, misconfigurations, and exploitable vulnerabilities that could allow unauthorized access, privilege escalation, data breaches, or service disruptions.

The penetration test targeted key AWS services, including IAM, EC2, S3, CloudFront, Lambda, and CloudTrail, focusing on potential attack vectors such as weak access controls, misconfigured permissions, publicly exposed resources, and insecure API endpoints. The results of this assessment provide a clear roadmap for strengthening AWS security defenses.

Scope:

The penetration test included a security evaluation of the following AWS services:

IAM – Privilege escalation risks, misconfigured policies, and missing MFA.
EC2 – Open ports, weak SSH/RDP access, and vulnerability exploitation.
S3 – Public exposure, insecure ACLs, and data leakage risks.
CloudFront – SSL/TLS misconfigurations and CORS vulnerabilities.
Lambda – Over-permissive roles, function abuse, and insecure API access.
CloudTrail & CloudWatch – Logging and monitoring gaps that could aid stealth attacks.

Methodology

The penetration testing process followed a structured approach:

- Reconnaissance & Enumeration – Identified publicly accessible services, open ports, and misconfigurations.
- Automated & Manual Exploitation – Used tools like Prowler and ScoutSuite, followed by manual validation to confirm exploitable weaknesses.
- Privilege Escalation & Lateral Movement – Tested IAM policies and misconfigurations that could allow unauthorized privilege escalation.
- Post-Exploitation & Impact Analysis – Evaluated the potential consequences of identified vulnerabilities.
- Risk Categorization & Reporting – Findings were classified as Critical, High, Medium, or Low, with detailed remediation steps.

This penetration test provides a comprehensive understanding of security risks, allowing proactive remediation to prevent potential breaches.

Overall Review summary

Vulnerability Assessment and Penetration Tester of the Web Application revealed weaknesses.
We observed that out of total **26** gaps; **5 Critical** Vulnerabilities were Found.

Refer to the tabular summary and pie chart of severity wise vulnerabilities identified during testing activity:

Severity/Risk	Count	% Of Total Count	Open
Critical	5	NA	0
High	3	NA	0
Medium	18	NA	0
Low	0	NA	0
Total Count	**26**	**NA**	**0**

Severity Metrics

Critical	Critical severity issues allow an attacker run arbitrary code on the underlying platform with the user's privileges in the normal course of usage. The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable and there is no acceptable alternative method to achieve the required results then the severity will be stated as critical.
High	These findings identify conditions that could directly result in the compromise or unauthorized access of a network, system, application or information. Examples of High Risks include known buffer overflows, weak or no passwords, no encryption, which could result in denial of service on critical systems or services; unauthorized access; and disclosure of information.
Medium	These findings identify conditions that do not immediately or directly result in the compromise or unauthorized access of a network, system, application or information, but do provide a capability or information that could, in combination with other capabilities or information, result in the compromise or unauthorized access of a network, system, application or information. Examples of Medium Risks include unprotected systems, files, and services that could result in denial of service on non-critical services or systems; and exposure of configuration information and knowledge of services or systems to further exploit.
Low	These findings identify conditions that do not immediately or directly result in the compromise of a network, system, application, or information, but do provide information that could be used in combination with other information to gain insight into how to compromise or gain unauthorized access to a network, system, application or information. Low risk findings may also demonstrate an incomplete approach to or application of security measures within the environment. Examples of Low Risks include cookies not marked secure; concurrent sessions and revealing system banners.

Detailed Findings

1. Hardcoded Secrets Found in Lambda Function Variables

Status: Close
Severity: Critical
Service Name: AWS Lambda
Region: us-east-1
Check ID: `awslambda_function_no_secrets_in_variables`
Check Title: Find secrets in Lambda function variables
Resource ID: [redacted]

Observation:

Potential secrets were detected in the **Lambda function `AtsResumeUploader`** environment variables:

- **AWS Access Key** in variable `s3accesskey`
- **AWS Access Key** in variable `sesconfig_awssecretaccesskey`
- **Secret Keyword** in variable `sesconfig_awssecretaccesskey`
- **Secret Keyword** in variable `postgresdb_dataSource_password`

Impact:

Hardcoded credentials increase the risk of **unauthorized access and credential leakage**. Attackers gaining access to exposed AWS access keys or database credentials can:

- Escalate privileges within the AWS environment.
- Exfiltrate sensitive data stored in S3, RDS, or DynamoDB.
- Deploy unauthorized compute resources, leading to financial and security risks.

Mitigation:

1. **Remove hardcoded secrets from Lambda function environment variables.**
2. **Use AWS Secrets Manager** to securely store database credentials and access keys.
3. **Configure IAM permissions** to ensure the Lambda function can access secrets without embedding them in the code.
4. **Enable AWS CloudTrail** to monitor access and detect potential abuse.

Reference:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html

---

2. Hardcoded Secrets Found in Lambda Function Variables

Status: Close
Severity: Critical
Service Name: AWS Lambda
Region: us-east-1
Check ID: `awslambda_function_no_secrets_in_variables`
Check Title: Find secrets in Lambda function variables
Resource ID: [redacted]

Observation:

Potential secrets were detected in the **Lambda function `jobboard_executor`** environment variables:

- **AWS Access Key** in variable `s3accesskey`
- **AWS Access Key** in variable `sesconfig_awssecretaccesskey`
- **Secret Keyword** in variable `sesconfig_awssecretaccesskey`
- **Secret Keyword** in variable `postgresdb_dataSource_password`

Impact:

The use of hardcoded credentials **significantly increases the risk of password exposure** and unauthorized access. Attackers may exploit these exposed credentials to:

- Gain unauthorized access to AWS services.
- Manipulate or delete cloud-based resources.
- Bypass security controls implemented at the IAM level.

Mitigation:

1. **Store all sensitive credentials in AWS Secrets Manager** instead of Lambda environment variables.
2. **Use IAM Roles** with the **least privilege** model to restrict access.
3. **Monitor access key usage** via AWS CloudTrail to detect unauthorized access.
4. **Rotate credentials periodically** to minimize exposure risk.

Reference:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html

---

3. Hardcoded Secrets Found in Lambda Function Variables

Status: Close
Severity: Critical
Service Name: AWS Lambda
Region: us-east-1
Check ID: `awslambda_function_no_secrets_in_variables`
Check Title: Find secrets in Lambda function variables
Resource ID: [redacted]

Observation:

Potential secrets were detected in the **Lambda function** [redacted] environment variables:

- **Secret Keyword** in variable [redacted]
- **Secret Keyword** in variable [redacted]
- **AWS Access Key** in variable [redacted]
- **Secret Keyword** in variable [redacted]

Impact:

Exposing credentials in environment variables **makes them vulnerable to accidental leaks or malicious access**. Attackers exploiting these secrets may:

- Gain **unauthorized access** to AWS resources.
- Execute privileged actions without detection.
- Compromise data stored in databases, object storage, or search services.

Mitigation:

1. **Remove secrets from environment variables** and store them in AWS Secrets Manager.
2. **Grant IAM permissions to Lambda functions** instead of embedding credentials.
3. **Regularly scan for secrets** in your AWS environment using automated tools like AWS Security Hub.

Reference:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html

---

4. Hardcoded Secrets Found in Lambda Function Variables

Status: Close
Severity: Critical
Service Name: AWS Lambda
Region: us-east-1
Check ID: `awslambda_function_no_secrets_in_variables`
Check Title: Find secrets in Lambda function variables
Resource ID: [redacted]

Observation:

Potential secrets were detected in the **Lambda function** environment variables:

- **Secret Keyword** in variable `s3secret`
- **AWS Access Key** in variable `s3accesskey`

Impact:

Embedding AWS secrets in environment variables **poses a security risk**, as they can be extracted by unauthorized users or malicious insiders. Attackers can:

- **Access AWS storage services** without authentication.
- **Modify or delete sensitive data** stored in S3 buckets.
- **Circumvent existing security controls** by using valid credentials.

Mitigation:

1. **Store credentials in AWS Secrets Manager** instead of embedding them in Lambda functions.
2. **Use IAM roles with fine-grained permissions** to control access to AWS resources.
3. **Rotate AWS keys periodically** to limit exposure.

Reference:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html

5. Hardcoded Secrets Found in Lambda Function Variables

Status: Close
Severity: Critical
Service Name: AWS Lambda
Region: us-east-1
Check ID: `awslambda_function_no_secrets_in_variables`
Check Title: Find secrets in Lambda function variables
Resource ID: [redacted]

Observation:

Potential secrets were detected in the **Lambda function `fastprofileattach`** environment variables:

- **AWS Access Key** in variable `sesconfig_awsaccesskeyid`
- **Secret Keyword** in variable `s3config_awssecretaccesskey`
- **Secret Keyword** in variable `postgresdb_dataSource_username`
- **AWS Access Key** in variable `s3config_awssecretaccesskey`
- **Secret Keyword** in variable `elasticsearch_dataSource_passwordprotected`
- **Secret Keyword** in variable `serverurldata_fastprofileloginpassword`

Impact:

Hardcoded secrets in Lambda functions **increase the risk of privilege escalation, unauthorized access, and data exfiltration**.

Mitigation:

1. **Use AWS Secrets Manager** to securely store credentials.
2. **Limit IAM permissions** to prevent over-privileged Lambda functions.
3. **Monitor API calls** using CloudTrail to detect unauthorized usage.

Reference:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html

6.EC2 Security Group Allows Ingress from the Internet to Any Port

Status: Close
Severity: High
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2_securitygroup_allow_ingress_from_internet_to_any_port`
Check Title: Ensure no security groups allow ingress from 0.0.0.0/0 or ::/0 to any port
Resource ID: [redacted]
Security Group Name: `supporting_server`

Observation:

The security group [redacted] has at least **one port open to the Internet**, exposing AWS resources to **unauthorized remote access**.

Impact:

This configuration can:

- Allow **unrestricted access** from malicious actors.
- Increase the risk of **DDoS attacks**.
- Lead to **unauthorized lateral movement** in AWS infrastructure.

Mitigation:

1. Restrict security group rules to **only trusted IPs**.
2. Implement **AWS Web Application Firewall (WAF)**.
3. Use **AWS Shield** for protection against DDoS attacks.

7.EC2 Security Group Allows Wide-Open Public IP Ingress

Status: Close
Severity: High
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2_securitygroup_allow_wide_open_public_ipv4`
Check Title: Ensure no security groups allow ingress and egress from wide-open IP address with a mask between 0 and 24
Resource ID: [redacted]
Security Group Name: `pg_server_group`

Observation:

The security group `sg-0ef63e3221f7c0` has an **open ingress rule for 000.45.0.0/16**, allowing unrestricted access from a **non-RFC18 address**.

Impact:

An overly permissive security group:

- Increases exposure to **brute-force attacks**.
- Allows attackers to **scan and exploit** open services.
- Violates **least privilege access principles**.

Mitigation:

1. Restrict ingress rules to **only necessary IPs**.
2. Implement **VPC Peering** or **AWS PrivateLink** for secure connectivity.
3. Regularly audit and **remove unused security groups**.

8.EC2 Security Group Allows Wide-Open Public IP Ingress

Status: Close
Severity: High
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2_securitygroup_allow_wide_open_public_ipv4`
Check Title: Ensure no security groups allow ingress and egress from wide-open IP address with a mask between 0 and 24
Resource ID: [redacted]
Security Group Name: [redacted]

Observation:

The security group s[redacted] allows inbound traffic from a **wide-open IP range (15.5.0.0/16)**, significantly increasing exposure to unauthorized access attempts.

Impact:

A permissive security group rule can:

- Expose AWS resources to **unauthorized access**.
- Allow attackers to **probe and exploit** open services.
- Increase the risk of **lateral movement within AWS infrastructure**.

Mitigation:

1. Restrict ingress rules to **only trusted IPs**.
2. Implement **least privilege** by allowing access to **specific subnets**.
3. Use **AWS Web Application Firewall (WAF)** for additional security.

9.EC2 Network ACL Allowing Ingress to SSH (Port 22)

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2_networkacl_allow_ingress_tcp_port_22`
Check Title: Ensure no Network ACLs allow ingress from 0.0.0.0/0 to SSH port 22
Resource ID: [redacted]

Observation:

The Network ACL [redacted] has **port 22 (SSH) open** to the public internet, increasing the risk of **unauthorized access and potential system compromise**.

Impact:

Publicly accessible SSH ports can:

- Be targeted by **brute-force attacks**.
- Allow unauthorized access to AWS instances.
- Be exploited for **privilege escalation** attacks.

Mitigation:

1. Restrict SSH access to **trusted IP addresses** only.
2. Use **AWS Systems Manager Session Manager** instead of SSH where possible.
3. Implement **key-based authentication** and disable password-based SSH login.

10.EC2 Network ACL Allowing Ingress to Microsoft RDP (Port 3389)

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2_networkacl_allow_ingress_tcp_port_3389`
Check Title: Ensure no Network ACLs allow ingress from 0.0.0.0/0 to Microsoft RDP port 3389
Resource ID: [redacted]

Observation:

The Network ACL [redacted] has **port** [redacted] **(RDP) open** to the public internet, increasing the risk of **unauthorized remote access and brute-force attacks**.

Impact:

Publicly accessible RDP ports can:

- Allow attackers to **brute-force RDP credentials** and gain unauthorized access.
- Expose sensitive resources to **ransomware attacks**.
- Increase the risk of **unauthorized lateral movement** within the AWS environment.

Mitigation:

1. Restrict RDP access to **trusted IP addresses** only.
2. Use **AWS Systems Manager Session Manager** instead of RDP where possible.
3. Implement **Multi-Factor Authentication (MFA)** for remote access.

11. Internet-Facing EC2 Instance with Attached Instance Profile

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2_instance_internet_facing_with_instance_profile`
Check Title: Check for Internet-Facing EC2 Instances with Instance Profiles Attached
Resource ID: [redacted]
Instance Name: [redacted]
Public IP: [redacted]
Instance Profile: [redacted]

Observation:

The EC2 instance [redacted] is **publicly accessible** and has an **Instance Profile attached**, increasing the **risk of unauthorized IAM credential access**.

Impact:

If compromised, this instance could:

- **Expose IAM role credentials** via metadata API.
- **Allow unauthorized access** to other AWS services.
- **Be used for privilege escalation** within AWS.

Mitigation:

1. **Remove the public IP** and use **private VPC subnets**.
2. **Restrict IAM permissions** following **least privilege access**.
3. **Monitor API calls** using AWS CloudTrail.
4. **Use AWS IAM Access Analyzer** to detect risky IAM permissions.

Reference:

https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html

---

12. Internet-Facing EC2 Instance with Attached Instance Profile

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2_instance_internet_facing_with_instance_profile`
Check Title: Check for Internet-Facing EC2 Instances with Instance Profiles Attached
Resource ID: [redacted]
Instance Name: `ES-1`
Public IP: [redacted]
Instance Profile: [redacted]

Observation:

The EC2 instance [redacted] has an attached **IAM Instance Profile** while being **publicly accessible**, which could lead to **IAM credential exposure**.

Impact:

An attacker compromising this instance could:

- **Extract AWS IAM role credentials** and misuse them.
- **Gain unauthorized access** to AWS resources.
- **Escalate privileges** through IAM misconfigurations.

Mitigation:

1. **Restrict public access** using **private subnets** and **security groups**.
2. **Review IAM role policies** to enforce **least privilege access**.
3. **Monitor IAM role activity** via AWS CloudTrail.
4. **Disable metadata service (IMDSv1)** to prevent credential theft.

Reference:

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use.html

---

13. Internet-Facing EC2 Instance with Attached Instance Profile

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2_instance_internet_facing_with_instance_profile`
Check Title: Check for Internet-Facing EC2 Instances with Instance Profiles Attached
Resource ID: [redacted]
Instance Name: [redacted]
Public IP: [redacted]
Instance Profile: [redacted]

Observation:

The EC2 instance [redacted] is **internet-facing** with an **IAM Instance Profile attached**, creating a **high-risk scenario** for AWS credential leaks.

Impact:

A compromised instance could:

- **Expose AWS IAM credentials** stored in metadata.
- **Allow unauthorized AWS service access** via the IAM role.
- **Be used for lateral movement** across AWS services.

Mitigation:

1. **Move the instance to a private subnet** and remove public IP.
2. **Apply security group restrictions** to **limit SSH/RDP access**.
3. **Monitor AWS IAM role activity** using AWS CloudTrail.
4. **Disable instance metadata v1 (IMDSv1)** to prevent unauthorized credential access.

Reference:

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use.html

---

14. EC2 Instance with Open SSH Port (22) to Public Internet

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2_instance_ssh_open_to_world`
Check Title: Check for EC2 Instances with Open SSH (Port 22) to the Internet
Resource ID: : [redacted]
Instance Name: [redacted]
Public IP: [redacted]

Observation:

The EC2 instance [redacted] has **port 22 (SSH) open to the public internet**, increasing the risk of **brute-force attacks and unauthorized access**.

Impact:

Publicly accessible SSH ports can:

- **Be targeted by automated brute-force attacks**.
- **Allow unauthorized remote access** to AWS infrastructure.
- **Lead to privilege escalation** if credentials are compromised.

Mitigation:

1. **Restrict SSH access** to specific trusted IP addresses.
2. **Use AWS Systems Manager Session Manager** instead of SSH.
3. **Implement Multi-Factor Authentication (MFA)** for SSH login.
4. **Monitor SSH access logs** using AWS CloudTrail.

Reference:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/TroubleshootingInstancesConnecting.html

15. Internet-Facing EC2 Instance with Attached Instance Profile

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2_instance_internet_facing_with_instance_profile`
Check Title: [redacted]d
Resource ID: `arn::instance/i-09095850`
Instance Name: [redacted]
Public IP: -[redacted]
Instance Profile: [redacted]

Observation:

The EC2 instance [redacted]s **internet-facing** and has an **Instance Profile attached**, increasing the risk of **IAM credential exposure**.

Impact:

An attacker compromising this instance can:

- **Extract IAM credentials** via the instance metadata service.
- **Use the IAM role** to interact with other AWS services like **S3, RDS, or DynamoDB**.
- **Escalate privileges** if the IAM role has excessive permissions.

Mitigation:

1. **Restrict internet exposure** by **removing the public IP** and using a bastion host.
2. **Review IAM policies** to ensure **least privilege access**.
3. **Monitor IAM role usage** via AWS CloudTrail.
4. **Rotate IAM role credentials periodically** to minimize exposure risk.

Reference:

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html

---

16. Internet-Facing EC2 Instance with Attached Instance Profile

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2`[redacted]
Check Title: Check for Internet-Facing EC2 Instances with Instance Profiles Attached
Resource ID: `::instance/i-030315`
Instance Name: [redacted]
Public IP: –[redacted]
Instance Profile: [redacted]

Observation:

The EC2 instance [redacted] **publicly accessible** and has an **Instance Profile attached**, increasing the **risk of unauthorized access and privilege escalation**.

Impact:

Attackers gaining access to this instance can:

- **Extract AWS credentials** stored in the metadata service.
- **Use the IAM role** to perform unauthorized actions within AWS.
- **Compromise other AWS services** linked to this role.

Mitigation:

1. **Remove the public IP** and use **private access via VPN or bastion host**.
2. **Limit IAM role permissions** to **only required actions**.
3. **Monitor role usage and API calls** using AWS CloudTrail.
4. **Enable MFA for IAM roles** wherever applicable.

Reference:

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html

17. EC2 Instance Exposed to Public Internet

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2_instance_public_ip`
Check Title: Check for EC2 Instances with Public IP
Resource ID: [redacted]
Instance Name: `PG-Replica`
Public IP: [redacted]

Observation:

The EC2 instance [redacted] is **publicly accessible**, exposing potential **database replication traffic** to external threats.

Impact:

Exposing a database replica instance to the internet can:

- **Allow unauthorized access** to sensitive backup data.
- **Lead to data corruption** if malicious actors manipulate replication settings.
- **Expose authentication credentials** to brute-force attacks.

Mitigation:

1. **Move the instance to a private VPC subnet** and remove the public IP.
2. **Use AWS Security Groups** to limit access to trusted instances only.
3. **Encrypt data in transit** with SSL/TLS for secure replication.
4. **Monitor replication logs** for suspicious activities.

Reference:

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ReadRepl.html

---

18. Internet-Facing EC2 Instance with Attached Instance Profile

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check ID:[redacted]
Check Title: Check for Internet-Facing EC2 Instances with Instance Profiles Attached
Resource ID: [redacted]
Instance Name:[redacted]
Public IP:`.compute-1.amazonaws.com`)
Instance Profile [redacted]

Observation:

The EC2 instance [redacted] is **internet-facing** and has an **Instance Profile attached**. This increases the **risk of credential leakage and privilege escalation** if compromised.

Impact:

An exposed EC2 instance with an attached IAM role can:

- **Be exploited by attackers** to extract IAM credentials.
- **Allow privilege escalation** if misconfigured permissions exist.
- **Provide access to other AWS resources** through lateral movement.

Mitigation:

1. **Restrict access to the instance** using **private IPs** and a **bastion host**.
2. **Detach unnecessary IAM roles** from internet-facing instances.
3. **Limit IAM role permissions** following the principle of **least privilege**.
4. **Use AWS CloudTrail** to monitor API calls made using the IAM role.

Reference:

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html

---

19. Internet-Facing EC2 Instance with Attached Instance Profile

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check I[redacted]
Check Title: Check for Internet-Facing EC2 Instances with Instance Profiles Attached
Resource ID: `arn`[redacted]
Instance Name:[redacted]
Public IP:`.compute-1.amazonaws.com`)
Instance Profile: [redacted]

Observation:

The EC2 instance [redacted] **internet-facing** and has an **Instance Profile attached**, increasing the risk of **unauthorized access and credential theft**.

Impact:

An attacker compromising this instance can:

- **Extract AWS credentials** from the instance metadata service.
- **Use the IAM role** to access AWS services such as **S3, RDS, or DynamoDB**.
- **Escalate privileges** if the attached role has excessive permissions.

Mitigation:

1. **Restrict internet exposure** by **removing the public IP** and using a bastion host.
2. **Review IAM policies** attached to the instance profile and enforce **least privilege access**.
3. **Monitor IAM role usage** using AWS CloudTrail.
4. **Rotate IAM role credentials periodically** to reduce exposure risk.

Reference:

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html

20. EC2 Instance Exposed to Public Internet

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2_instance_public_ip`
Check Title: Check for EC2 Instances with Public IP
Resource ID: [redacted]
Instance Name: `PG-Replica`
Public IP: [redacted]

Observation:

The EC2 instance [redacted] is **publicly accessible**, exposing potential **database replication traffic** to external threats.

Impact:

Exposing a database replica instance to the internet can:

- **Allow unauthorized access** to sensitive backup data.
- **Lead to data corruption** if malicious actors manipulate replication settings.
- **Expose authentication credentials** to brute-force attacks.

Mitigation:

1. **Move the instance to a private VPC subnet** and remove the public IP.
2. **Use AWS Security Groups** to limit access to trusted instances only.
3. **Encrypt data in transit** with SSL/TLS for secure replication.
4. **Monitor replication logs** for suspicious activities.

Reference:

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ReadRepl.html

21. EC2 Instance Exposed to Public Internet

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2_instance` [redacted]
Check Title: Check for EC2 instances with Public IP
Resource ID: `arn:aws:ec2`
Instance Name: [redacted]
Public IP: [redacted]

Observation:

The EC2 instance [redacted] is **publicly accessible**, increasing the attack surface and potential security risks.

Impact:

Public-facing EC2 instances can be exploited by attackers to:

- **Perform unauthorized access attempts** via open services.
- **Execute remote code attacks** if vulnerabilities exist.
- **Compromise application or database servers** hosted on the instance.

Mitigation:

1. **Remove the public IP** and use a **bastion host** or **VPN** for remote access.
2. **Implement AWS Security Groups** to restrict inbound traffic.
3. **Use AWS WAF & ALB** to protect against external threats.
4. **Enable CloudTrail logs** to monitor unauthorized access attempts.

Reference:

https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Security.html

---

22. EC2 Instance Exposed to Public Internet

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check ID: [redacted]
Check Title: Check for EC2 Instances with Public IP
Resource ID: [redacted]116250d
Instance Name: `PG_SERVER`
Public IP: [redacted]

Observation:

The EC2 instance [redacted] has a **public IP assigned**, making it **directly accessible over the internet**. This increases the risk of **unauthorized access and credential exposure**.

Impact:

A publicly exposed PostgreSQL database server could:

- **Allow brute-force attacks** on database credentials.
- **Expose critical data** if misconfigured or unencrypted.
- **Be targeted for SQL injection** or other exploits.

Mitigation:

1. **Use AWS Secrets Manager** to securely store credentials.
2. **Restrict database access** to **internal VPC** or **specific IPs only**.
3. **Enable TLS encryption** to secure database connections.
4. **Monitor database logs** for suspicious access patterns.

Reference:

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.DBInstance.html

---

23. EC2 Instance Exposed to Public Internet

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2_instance_public_ip`
Check Title: Check for EC2 Instances with Public IP
Resource ID: [redacted]
Instance Name: `ES-1`
Public IP: [redacted]

Observation:

The EC2 instance [redacted] is exposed to the internet via a **public IP address**, increasing the **risk of unauthorized access and potential compromise**.

Impact:

Publicly exposed instances can:

- **Be exploited by attackers** to gain unauthorized access.
- **Serve as an entry point** for lateral movement across the AWS environment.
- **Become a target for botnet attacks** or automated scanning.

Mitigation:

1. **Remove the public IP** and use a **bastion host** for SSH access.
2. **Apply strict security group rules** to restrict access.
3. **Enable logging & monitoring** using AWS CloudWatch.
4. **Deploy AWS Shield** to mitigate potential DDoS attacks.

Reference:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/security-best-practices.html

24. EC2 Instance Exposed to Public Internet

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2_instance_public_ip`
Check Title: Check for EC2 Instances with Public IP
Resource ID: [redacted]
Instance Name: `ES-2`
Public IP: [redacted]

Observation:

The EC2 instance [redacted] is publicly accessible, exposing it directly to the internet. This increases the attack surface and poses a risk of **unauthorized access, brute-force attacks, and exploitation of vulnerabilities**.

Impact:

Publicly accessible EC2 instances can be targeted by attackers to:

- Exploit open network services and system vulnerabilities.
- Launch **DDoS** or **brute-force attacks** against exposed services.
- Compromise sensitive data stored or processed within the instance.

Mitigation:

1. **Restrict public access** by removing the public IP and using a private IP with a **bastion host**.
2. **Implement AWS Application Load Balancer (ALB)** with **Web Application Firewall (WAF)** for security.
3. **Use security groups and Network ACLs** to limit access to trusted IPs only.
4. **Enable AWS Shield** to protect against DDoS attacks.

Reference:

https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Security.html

---

25. EC2 Instance Exposed to Public Internet

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2_instance_public_ip`
Check Title: Check for EC2 Instances with Public IP
Resource ID: [redacted]
Instance Name: `app`
Public IP [redacted]

Observation:

The EC2 instance [redacted] is publicly accessible. This poses a **security risk of unauthorized access, brute-force attacks, and system compromise**.

Impact:

Publicly exposed EC2 instances increase the risk of:

- **Credential stuffing attacks** against open services.
- **Data breaches** if sensitive data is stored on the instance.
- **Malware infections** if security patches are outdated.

Mitigation:

1. **Assign a private IP** and use a **VPN or bastion host** for secure access.
2. **Configure AWS Security Groups** to allow inbound traffic only from trusted IPs.
3. **Implement AWS WAF** to filter and monitor incoming traffic.
4. **Disable unused services** running on the instance.

Reference:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-instance-addressing.html

---

26. EC2 Instance Exposed to Public Internet

Status: Exception
Severity: Medium
Service Name: Amazon EC2
Region: us-east-1
Check ID: `ec2_instance_public_ip`
Check Title: Check for EC2 Instances with Public IP
Resource ID: [redacted]
Instance Name: `Mongo`
Public IP: [redacted]

Observation:

The EC2 instance [redacted] has a **public IP assigned**, making it **directly accessible over the internet**. This increases the risk of **unauthorized access, exploitation, and data theft**.

Impact:

Exposing a MongoDB instance to the internet could:

- **Allow unauthorized users to query or modify databases.**
- **Expose authentication credentials** to brute-force attacks.
- **Be a target for ransomware** or database wiping attacks.

Mitigation:

1. **Restrict access** by removing the public IP and using a **private VPC endpoint**.
2. **Apply IAM-based access controls** for database interactions.
3. **Enable encryption** for data at rest and in transit.
4. **Regularly audit access logs** using AWS CloudTrail.

Reference:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/security-best-practices.html

Conclusion

The security assessment of the AWS infrastructure has revealed multiple vulnerabilities, including **misconfigured security groups, open network ACLs, and publicly exposed EC2 instances**. These weaknesses increase the risk of **unauthorized access, brute-force attacks, and system compromise**.

Key issues include **security groups allowing unrestricted internet access**, **publicly exposed EC2 instances**, and **open SSH/RDP ports**, which could be exploited for **unauthorized entry, remote code execution, or data breaches**.

Recommended Actions:

1. **Restrict Security Group Ingress** – Allow access only from **trusted IPs**.
2. **Harden Network ACLs** – Block unrestricted inbound traffic to **SSH (22) and RDP (3389)**.
3. **Disable Public EC2 Access** – Use **bastion hosts or VPNs** instead.
4. **Enable AWS Security Services** – Use **AWS WAF, Shield, and GuardDuty** for continuous monitoring.

Implementing these measures will **reduce the attack surface and strengthen security**. Regular **audits and automated compliance checks** are recommended to ensure ongoing protection.

Disclaimer:

This report is for **authorized personnel only** and must not be shared or disclosed without prior consent. Unauthorized use is strictly prohibited.