



{Client Name}  
{Month} {Year}

# {Client Name} AWS Security Configuration Review Report

### Client Details

Company Name: {Client Name}  
Contact Person: {Person Name}  
Address: {Address Name}  
Email: {Email Address}  
Telephone: {Telephone Number}

### Document History

Version	Date	Author	Remark
1.0	{Date}	{Author}	Document Creation

## TABLE OF CONTENTS

1. Introduction.....	3
2. Summary .....	3
3. Summary Table.....	3
4. Details .....	5
4.1 IAM.....	5
4.1.1 Policies.....	5
1) Managed Policy with IAM:PassRole .....	5
2) Inline Role Policy with IAM:PassRole .....	6
3) Inline User Policy with IAM:PassRole .....	7
4.1.2 Access Keys.....	8
1) Lack of Key Rotation .....	8
2) A user with multiple Access Key .....	9
4.1.2 Users .....	10
1) Root User Account Recently Used.....	10
2) Multi-Factor Authentication.....	11
3) Password Policy .....	12
4.2 RDS.....	12
4.2.1 Auto-Minor Version Upgrade .....	12
4.2.2 Backup Retention Period .....	13
4.3 Redshift.....	14
4.3.1 Cluster Database Encryption .....	14
4.3.2 Clusters are publicly accessible .....	15
4.3.3 Group Parameter – SSL required.....	16
4.3.4 Group Parameter – User Activity Logging.....	17
4.4 S3.....	18
4.4.1 World Writable Buckets – All Users .....	18
4.4.2 World Readable and Writable Buckets – Authenticated Users .....	19
4.4.3 World Readable Buckets – All Users .....	20
4.4.4 Logging.....	22
4.4.5 Versioning .....	23
4.4.6 Versioned bucket without MFA delete.....	24
4.5 EC2 .....	25
4.5.1 Ports open to all .....	25
4.5.2 Default Security Groups .....	26

## 1. Introduction

This document details the findings of the security configuration review done on AWS. The following analysis and recommendations on securing AWS are backed up by our vast field experience with managing secure server installations, forensics study done on compromised server machines, and the leading international security standards and best practices of Amazon Web Services security recommendations. The policies that do not conform to the recommendations are highlighted and explained in the “Details” section of this document.

## 2. Summary

From the security point of view it looks like few of the security configuration part has been taken care during the initial setup of the server but most of the critical aspect like a password policy, managed policy, RDS, Redshift, EC2 security groups, and S3 configuration has not been configured/installed from a security perspective.

## 3. Summary Table

Colour legend:

Red	High Risk
Yellow	Medium Risk
Blue	Low Risk
White	Informational

S. No.	Policy/Parameter	Current Setting	Recommended Setting
<b>1.</b>	<b>IAM</b>		
1.1	Policy		
1.1.1	Managed policy iam:PassRole	Enabled	Disabled
1.1.2	Inline role policy iam:PassRole	Enabled	Disabled
1.1.3	Inline user policy iam:PassRole	Enabled	Disabled
1.2	Access Keys		
1.2.1	Key rotation	Disabled	Enabled
1.2.2	A user with multiple API keys	Allowed	Not Allowed
1.3	Users		
1.3.1	Root User Active	Enabled	Disabled
1.3.2	Multi-Factor Authentication	Disabled	Enabled
1.3.3	Password Policy	Disabled	Enabled

<b>2.</b>	<b>RDS</b>		
2.1	Auto-Minor Version Upgrade	Disabled	Enabled
2.2	Backup Retention Period	7/14 Days	30 Days (Based on business requirement)
<b>3.</b>	<b>Redshift</b>		
3.1	Required SSL	Disabled	Enabled
3.2	Cluster Database Encryption	Disabled	Enabled
3.3	Cluster publicly accessible	Enabled	Disabled
3.4	User Activity Logging	Disabled	Enabled
<b>4.</b>	<b>S3 Buckets</b>		
4.1	World Writable – All Users	Enabled	Disabled
4.2	World Readable – All Users	Enabled	Disabled
4.3	World Writable and Readable – Authenticated Users	Enabled	Disabled
4.4	Versioning	Disabled	Enabled
4.5	Logging	Disabled	Enabled
4.6	Versioned bucket without MFA delete	Disabled	Enabled
4.7	Static Website	Enabled	Based on business requirement
<b>5.</b>	<b>EC2</b>		
5.1	Security Groups		
5.1.1	Ports and Services Are Open To All	Enabled	Disabled
5.1.2	Default Security Group Are Used	Enabled	Based on business requirement

## 4. Details

This section provides the details of the configurations that do not conform to the recommended settings. It explains the threats and risks associated with non-conforming policies.

### 4.1 IAM

#### 4.1.1 Policies

##### 1) Managed Policy with IAM:PassRole

Managed policies are used iam:PassRole with \* as the value of resource element. It means the The policy will allow the IAM role to pass any roles mentioned in the Resource section to the EC2 instance.

This will allow a user or service with a low privilege role to assume a high privilege role and thus escalate the privilege of the restricted user.

Following managed policies are using iam:PassRole action.

1. AWSElasticBeanstalkServiceRolePolicy
2. AmazonEC2SpotFleetRole
3. AWSOpsWorksRegisterCLI
4. AmazonEC2ContainerServiceFullAccess
5. AWSElasticBeanstalkService

Following image shows the details of “AWSElasticBeanstalkService” used.

```

File Edit View Search Terminal Help
"elasticloadbalancing:ConfigureHealthCheck",
"elasticloadbalancing:CreateLoadBalancer",
"elasticloadbalancing>DeleteLoadBalancer",
"elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
"elasticloadbalancing:DescribeInstanceHealth",
"elasticloadbalancing:DescribeLoadBalancers",
"elasticloadbalancing:DescribeTargetHealth",
"elasticloadbalancing:RegisterInstancesWithLoadBalancer",
"elasticloadbalancing:DescribeTargetGroups",
"elasticloadbalancing:RegisterTargets",
"elasticloadbalancing:DeregisterTargets",
"iam:ListRoles",
"iam:PassRole",
"logs:CreateLogGroup",
"logs:PutRetentionPolicy",
"rds:DescribeDBEngineVersions",
"rds:DescribeDBInstances",
"rds:DescribeOrderableDBInstanceOptions",
"s3:CopyObject",
"s3:GetObject",
"s3:GetObjectAcl",
"s3:GetObjectMetadata",
"s3:ListBucket",
"s3:ListBuckets",
"s3:ListObjects",
"sns:CreateTopic",
"sns:GetTopicAttributes",
"sns:ListSubscriptionsByTopic",
"sns:Subscribe",
"sns:SetTopicAttributes",
"sqs:GetQueueAttributes",
"sqs:GetQueueUrl",
"codebuild:CreateProject",
"codebuild>DeleteProject",
"codebuild:BatchGetBuilds",
"codebuild:StartBuild"
],
"Resource": [
  "*"
],
"Effect": "Allow",
"Sid": "AllowOperations"
}
},
"IsDefaultVersion": true
}
}

```

iam:PassRole with resource \*

**Recommendation-** If possible, try to avoid the use of IAM:PassRole; if not, implement a filter to restrict the iam:PassRole permission with the Resources element of the IAM policy statement.

**Reference** - [http://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_roles\\_use\\_passrole.html](http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_passrole.html)

## 2) Inline Role Policy with IAM:PassRole

Role Inline policy has iam:PassRole with \* as a value of resource element. It means the The policy will allow the IAM role to pass any roles mentioned in the Resource section to the EC2 instance.

This will allow a user or service with a low privilege role to assume a high privilege role and thus escalate the privilege of the restricted user.

Following inline role policies are using iam:PassRole action.

1. aws-opsworks-service-policy

Following is a screenshot of “aws-opsworks-service-policy” with iam:PassRole

AWS Security Configuration Review Report	Document Level: Confidential
Document Version: 1.0	Page 6 of 28

```

$ aws --profile [redacted] iam get-user-policy --user-name [redacted] --policy-name iam-PassRole
{
  "UserName": "[redacted]",
  "PolicyName": "iam-PassRole",
  "PolicyDocument": {
    "Version": "[redacted]",
    "Statement": [
      {
        "Action": [
          "iam:PassRole"
        ],
        "Resource": [
          "*"
        ],
        "Effect": "Allow",
        "Sid": "[redacted]464114452000"
      }
    ]
  }
}

```

**Recommendation-** If possible, try to avoid the use of IAM:PassRole; if not, implement a filter to restrict the iam:PassRole permission with the Resources element of the IAM policy statement.

**Reference -** [http://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_roles\\_use\\_passrole.html](http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_passrole.html)

### 3) Inline User Policy with IAM:PassRole

User Inline policy has iam:PassRole with \* as a value of resource element. It means the The policy will allow the IAM user to pass any roles mentioned in the Resource section to the EC2 instance.

This will allow a user or service with a low privilege role to assume a high privilege role and thus escalate the privilege of the restricted user.

Following inline user policies are using iam:PassRole action.

1. iam-PassRole
2. code-deploy

Following is a screenshot of “iam-PassRole” with iam:PassRole

```

$ aws --profile [redacted] iam get-user-policy --user-name [redacted] --policy-name iam-PassRole
{
  "UserName": "[redacted]",
  "PolicyName": "iam-PassRole",
  "PolicyDocument": {
    "Version": "[redacted]",
    "Statement": [
      {
        "Action": [
          "iam:PassRole"
        ],
        "Resource": [
          "*"
        ],
        "Effect": "Allow",
        "Sid": "[redacted]"
      }
    ]
  }
}

```

**Recommendation-** If possible try to avoid use of IAM:PassRole, if not implement a filter to restrict the iam:PassRole permission with the Resources element of the IAM policy statement.



**Reference** - [http://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_roles\\_use\\_passrole.html](http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_passrole.html)

#### **4.1.2 Access Keys**

##### **1) Lack of Key Rotation**

After reviewing IAM configuration, we found that some IAM users are using access keys that are not renewed or rotated even after 2 years, which implicates the key rotation policy is not implemented. Due to the absence of a key rotation policy, there are increased chances that a compromised set of access keys can be used without admin knowledge to access AWS services.

If an attacker somehow gets hold of the access key, he/she can access the AWS infrastructure services depending on the permissions tied to the compromised access key without admin knowledge till the key is manually invalidated. Access Key rotation policy will invalidate the keys after every configured number of days.

Following is the snapshot of users' and their keys with the date of creation and activation status.

AWS Security Configuration Review Report	Document Level: Confidential
Document Version: 1.0	Page 8 of 28

```
python iam-users-access-keys.py
```

Found 56 users

User	Keys	Created	Status
		2015-05-05 06:21:47+00:00	Active
		2015-08-14 04:14:54+00:00	Active
		2017-04-25 22:56:46+00:00	Active
		2014-11-24 19:14:22+00:00	Active
		2017-08-23 22:37:19+00:00	Active
		2016-06-26 10:08:43+00:00	Active
		2015-03-12 18:49:57+00:00	Active
		2015-10-28 03:08:30+00:00	Active
		2016-08-25 07:20:49+00:00	Active
		2016-07-08 17:21:35+00:00	Active
		2017-10-05 17:49:41+00:00	Active
		2016-03-13 04:21:47+00:00	Active
		2016-05-11 11:27:10+00:00	Active
		2015-12-11 20:25:48+00:00	Active
		2015-11-19 19:19:02+00:00	Active
		2015-05-02 03:55:58+00:00	Active
		2016-07-07 17:02:01+00:00	Active
		2017-09-13 18:59:57+00:00	Active
		2017-08-28 15:58:29+00:00	Active
		2017-09-13 18:38:34+00:00	Active
		2017-08-01 14:34:22+00:00	Active
		2017-08-01 14:31:59+00:00	Active
		2017-09-13 18:39:30+00:00	Active
		2017-08-28 15:59:40+00:00	Active
		2017-09-13 19:01:07+00:00	Active
		2017-08-01 14:33:12+00:00	Active
		2017-09-13 19:02:37+00:00	Active
		2017-09-13 18:40:43+00:00	Active
		2017-08-28 16:00:22+00:00	Active
		2017-08-28 18:51:05+00:00	Active
		2017-08-28 18:58:29+00:00	Active
		2017-08-28 19:06:55+00:00	Active
		2017-08-28 19:18:49+00:00	Active
		2017-08-28 19:15:05+00:00	Active
		2015-08-01 08:25:53+00:00	Active
		2015-08-01 03:39:07+00:00	Active
		2016-07-02 17:00:00+00:00	Active
		2015-08-18 15:11:47+00:00	Active
		2016-05-11 11:22:13+00:00	Active
		2015-08-13 16:28:50+00:00	Active
		2017-08-28 18:39:55+00:00	Active
		2017-06-23 01:58:00+00:00	Active
		2014-11-25 19:49:17+00:00	Active
		2015-06-15 21:11:13+00:00	Active
		2014-11-26 01:48:06+00:00	Active
		2015-02-23 20:38:15+00:00	Active
		2015-02-23 20:38:15+00:00	Active
		2015-02-23 20:38:15+00:00	Active
		2015-02-23 20:38:15+00:00	Active
		2015-03-11 06:58:01+00:00	Active
		2015-08-01 03:39:06+00:00	Active
		2016-07-12 18:52:53+00:00	Active

**Recommendation** – Implement a key rotation policy for access keys. 45 days is the recommended value.

**Reference** - [http://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_credentials\\_access-keys.html#Using\\_RotateAccessKey](http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html#Using_RotateAccessKey)

## 2) A user with multiple Access Key

After reviewing IAM user configuration, we found that users are using two access keys. By configuring an account with multiple access keys will unintentionally introduce access flaws due to different permissions for different access keys belonging to the same account. It will make it difficult to keep track of the usage of keys if one of the keys will get compromised, and you will get an unclear audit trail.

Found 56 users

python iam-users-access-keys.py

User	Keys	Created	Status
		2015-05-05 06:21:47+00:00	Active
		2015-08-14 04:14:54+00:00	Active
		2017-04-25 22:56:46+00:00	Active
		2014-11-24 19:14:22+00:00	Active
		2017-08-23 22:37:19+00:00	Active
		2016-06-26 10:08:43+00:00	Active
		2015-03-12 18:49:57+00:00	Active
		2015-10-28 03:08:30+00:00	Active
		2016-08-25 07:20:49+00:00	Active
		2016-07-08 17:21:35+00:00	Active
		2017-10-05 17:49:41+00:00	Active
		2016-03-13 04:21:47+00:00	Active
		2016-05-11 11:27:10+00:00	Active
		2015-12-11 20:25:48+00:00	Active
		2015-11-19 19:19:02+00:00	Active
		2015-05-02 03:55:58+00:00	Active
		2016-07-02 17:02:01+00:00	Active
		2017-09-13 18:59:57+00:00	Active
		2017-08-28 15:58:29+00:00	Active
		2017-09-13 16:36:34+00:00	Active
		2017-08-01 14:34:22+00:00	Active
		2017-08-01 14:31:59+00:00	Active
		2017-09-13 18:39:30+00:00	Active
		2017-08-28 15:59:40+00:00	Active
		2017-09-13 19:01:07+00:00	Active
		2017-08-01 14:33:12+00:00	Active
		2017-09-13 19:02:37+00:00	Active
		2017-09-13 18:40:43+00:00	Active
		2017-08-28 16:00:22+00:00	Active
		2017-08-28 18:51:05+00:00	Active
		2017-08-28 18:58:29+00:00	Active
		2017-08-28 19:06:55+00:00	Active
		2017-08-28 19:18:49+00:00	Active
		2017-08-28 19:15:05+00:00	Active
		2015-08-01 08:25:53+00:00	Active
		2015-08-01 03:39:07+00:00	Active
		2016-07-02 17:00:00+00:00	Active
		2015-06-16 15:11:47+00:00	Active
		2016-05-11 11:22:13+00:00	Active
		2015-08-13 16:28:50+00:00	Active
		2017-08-28 18:39:55+00:00	Active
		2017-06-23 01:50:00+00:00	Active
		2014-11-25 19:49:17+00:00	Active
		2015-06-15 21:11:13+00:00	Active
		2014-11-26 01:48:06+00:00	Active
		2015-02-23 20:38:15+00:00	Active
		2015-02-23 20:38:15+00:00	Active
		2015-02-23 20:38:15+00:00	Active
		2015-02-23 20:38:15+00:00	Active
		2015-03-11 06:58:01+00:00	Active
		2015-08-01 03:39:06+00:00	Active
		2016-07-12 18:52:53+00:00	Active

**Recommendation** – Always use single access key for a user account and remove other keys that are not in use.

**Reference** - [http://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_credentials\\_access-keys.html#Using\\_RotateAccessKey](http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html#Using_RotateAccessKey)

## 4.1.2 Users

### 1) Root User Account Recently Used

After reviewing the credential report of IAM services, we found that the root user is recently used. The root user is recently logged into AWS dashboard using a password.

The root account is the most powerful account in AWS. It is recommended to lockout the root account because if the credentials of the root account get compromised, it can damage the whole AWS infrastructure setup, which in turn can damage the whole IoT ecosystem.



**Recommendation** – It is recommended to enable MFA for a privileged user to add extra security.

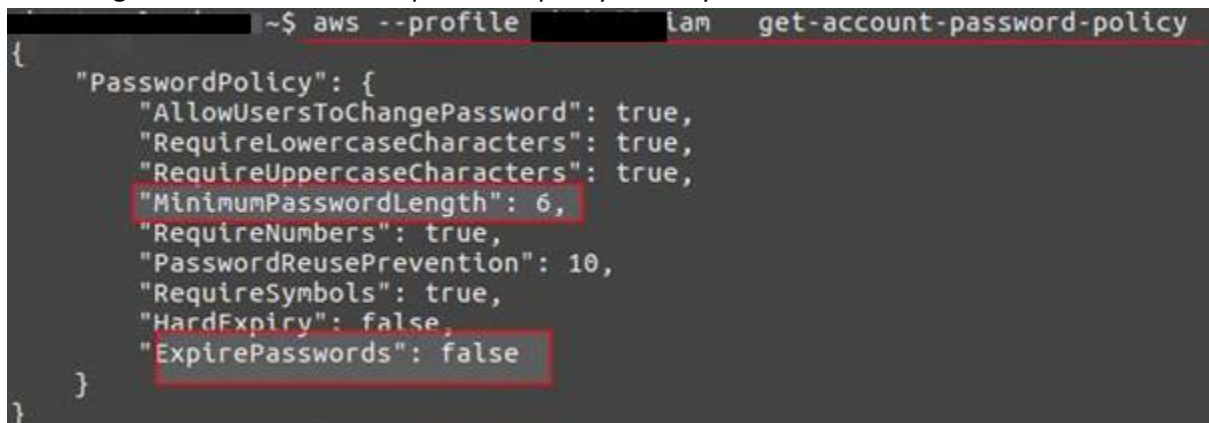
**Reference** - <http://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#enable-mfa-for-privileged-users>

### 3) Password Policy

After reviewing the IAM password policy, we found that the minimum length for a password is short, and the password expiration configuration is not set. It is easy for an attacker to brute-force a password with a short length. Password rotation policy protects against the usage of compromised passwords.

It will allow an attacker to figure out passwords of the AWS accounts easily by using brute-force attacks, and by using the figured-out passwords, the attacker can manipulate AWS infrastructure.

Following screenshot shows the password policy used by Client infrastructure.



```
~$ aws --profile [redacted] iam get-account-password-policy
{
  "PasswordPolicy": {
    "AllowUsersToChangePassword": true,
    "RequireLowercaseCharacters": true,
    "RequireUppercaseCharacters": true,
    "MinimumPasswordLength": 6,
    "RequireNumbers": true,
    "PasswordReusePrevention": 10,
    "RequireSymbols": true,
    "HardExpiry": false,
    "ExpirePasswords": false
  }
}
```

**Recommendation** – It is recommended to strong password policy by increasing the minimum length of the password and by setting password expiration.

**Reference** - <http://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#configure-strong-password-policy>

## 4.2 RDS

### 4.2.1 Auto-Minor Version Upgrade

After reviewing the configuration of RDS instances, we found that "Auto Minor Version Upgrade" setting is disabled. To receive automatic minor engine upgrades, Auto Minor Version Upgrade should be enabled. AWS upgrades database engine in a specified maintenance window.

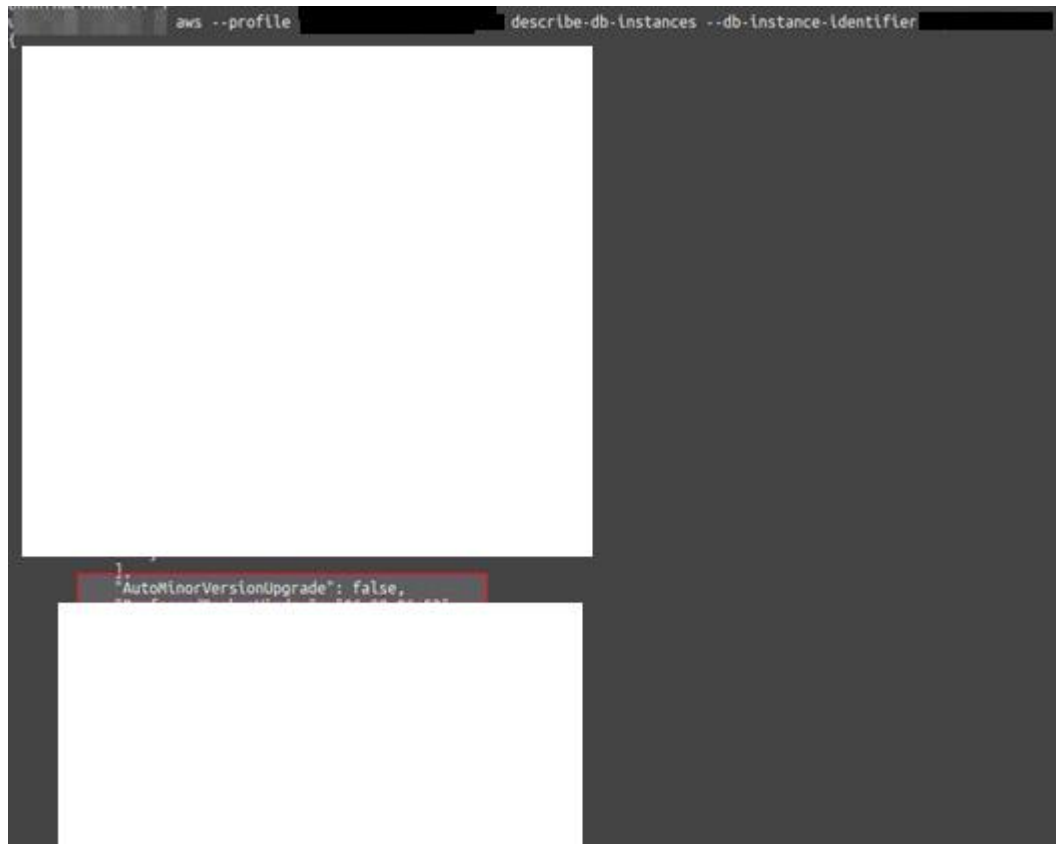
The absence of this configuration will not allow AWS to provide updates to the RDS database engine used.

Following RDS instance is affected.

1. <instance name>



Following screenshot shows the database configuration.



**Recommendation** – Enable “Auto Minor Version Upgrade” settings of RDS for receiving patches and updates for RDS engine in use.

**Reference** -

[http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER\\_UpgradeDBInstance.Upgrade.html](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_UpgradeDBInstance.Upgrade.html)

#### 4.2.2 Backup Retention Period

After reviewing the configuration of RDS instances, we found that "Backup Retention Period" is short. Retaining RDS backups for a longer period will allow you to handle more efficiently your data restoration/retention process in the event of failure.

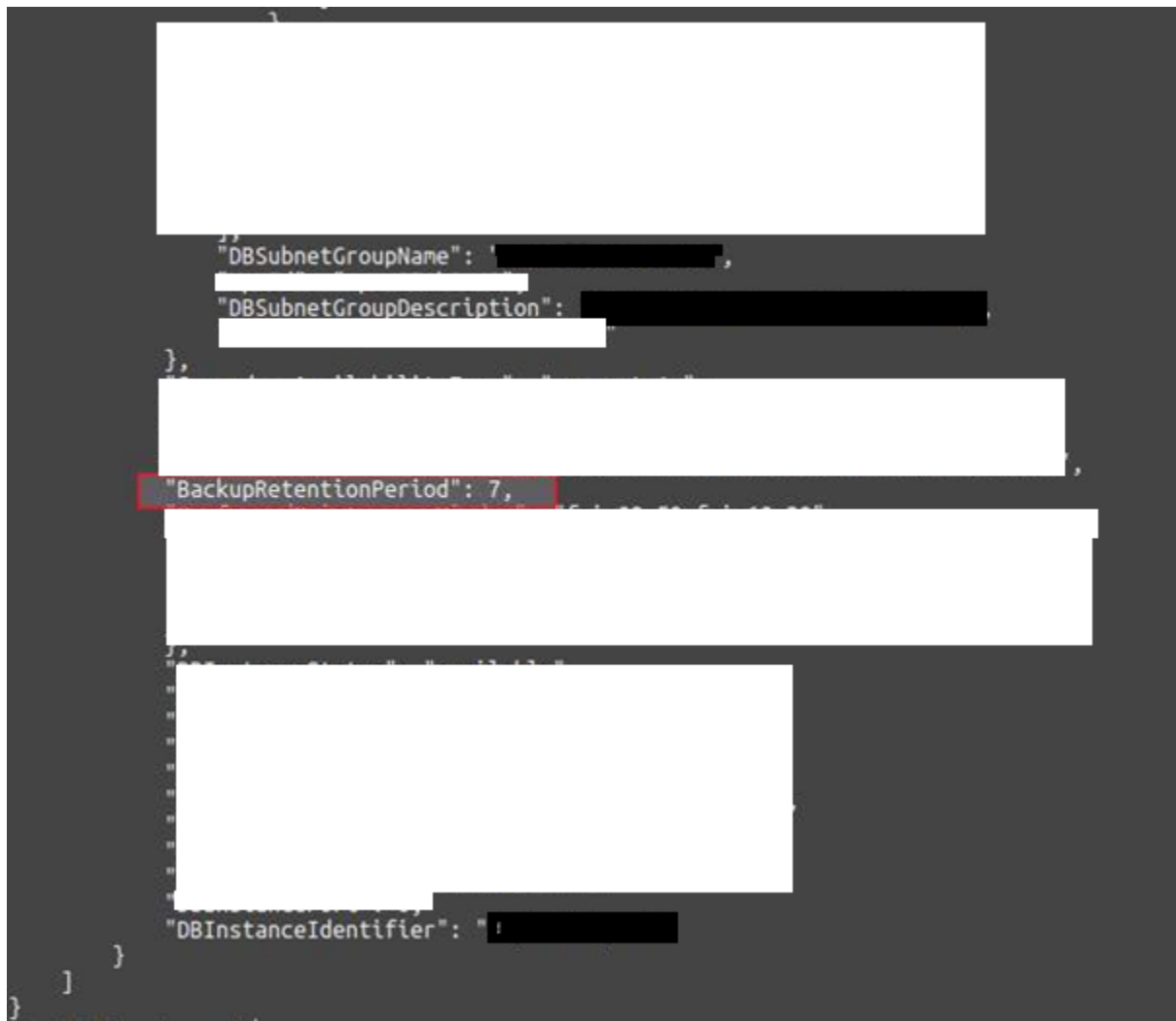
Due to the short backup retention period, AWS will not keep the backup files of the database for a longer duration, which makes the restoration of data difficult in the event of failure.

Following RDS instance is affected.

1. <instance name>

Following screenshot shows the database configuration.

AWS Security Configuration Review Report	Document Level: Confidential
Document Version: 1.0	Page 13 of 28



**Recommendation** – Set "Backup Retention Period" for more than 7 days. It will be completely dependent on the business process.

**Reference** -

[http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER\\_WorkingWithAutomatedBackups.html](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_WorkingWithAutomatedBackups.html)

## 4.3 Redshift

### 4.3.1 Cluster Database Encryption

After reviewing the configuration of <instances name> instances, we found that the "Database Encryption" setting is disabled. Due to this, databases with sensitive data will not be encrypted and are susceptible to unauthorized access.

The information within the database will be stored in plain text, which will allow an unauthorized user to get hold of sensitive data easily.

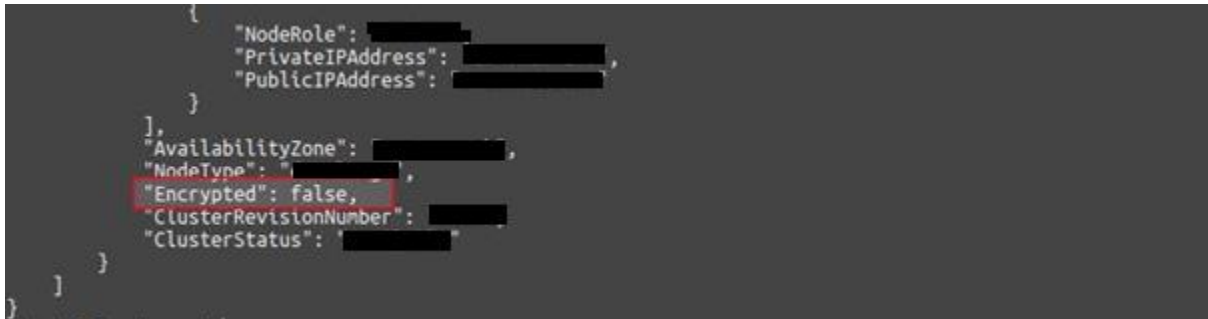
Following Cluster instance is affected.

1. test

AWS Security Configuration Review Report	Document Level: Confidential
Document Version: 1.0	Page 14 of 28

2. mobile-analytics
3. <xyz>-analytics

Following is the configuration of the “mobile-analytics” cluster.



**Recommendation** – Set “Encrypted” property of <cluster name> cluster to true in order to protect data at rest.

**Reference** - <http://docs.aws.amazon.com/redshift/latest/mgmt/working-with-db-encryption.html>

#### 4.3.2 Clusters are publicly accessible

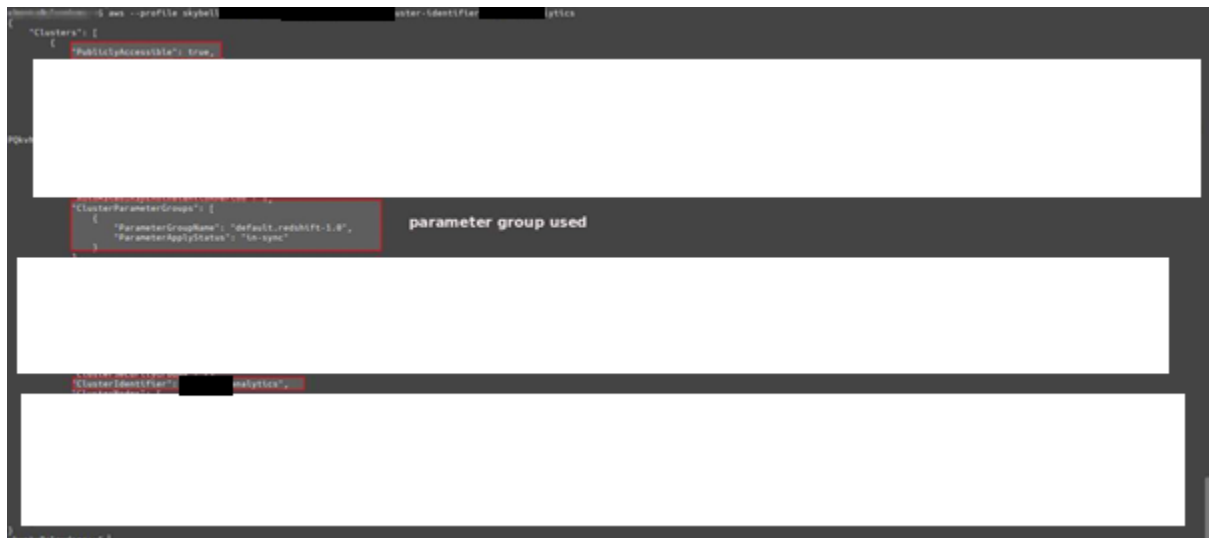
After reviewing the configuration of <Instance name> instances, we found that clusters are publicly accessible. When clusters are publicly accessible, any machine can access them and carry out malicious activities such as SQL injection or DDOS (Distributed Denial of Service) attacks, which will make service unavailable for IoT devices, users, apps, etc.

Following Cluster instance is affected.

1. test
2. mobile-analytics
3. <xyz>-analytics

Following is the configuration of “<xyz>-analytics”.





**Recommendation** – It is recommended to access the <cluster name> cluster from VPC only, and it is completely dependent on the business process

**Reference** - <http://docs.aws.amazon.com/redshift/latest/mgmt/managing-clusters-console.html>

### 4.3.3 Group Parameter – SSL required

After reviewing the configuration of the parameter group attached to Redshift instances, we found that “require\_ssl” is disabled, which makes communication between clients and these clusters susceptible to man in the middle attack.

Since the require\_ssl property is disabled, an attacker can intercept and modify the communication between AWS servers and service consumers such as IoT devices, mobile apps, etc.

Following parameter group instance is affected.

1. default.redshift-1.0

Following is the configuration of “<name>-analytics” with the attached parameter group “default-redshift-1.0”. Following is a screenshot of the parameter group configuration.



**Recommendation** – Set `require_ssl` parameter to `true`. This parameter will protect data in transit.

**Reference** - <http://docs.aws.amazon.com/redshift/latest/mgmt/working-with-parameter-groups.html>

#### 4.3.4 Group Parameter – User Activity Logging

After reviewing the configuration of the parameter group attached to Redshift instances, we found that “`enable_user_activity_logging`” is disabled due to this, AWS will not log user activities, access log, and queries performed on redshift cluster instances.

Any unauthorized activity (unauthorized access, attacks) will not be logged in the system, so in case of any security incident, there will be no log to analyze for finding the root cause and actor of the unauthorized activity.

Following parameter group instance is affected.

1. default.redshift-1.0

Following is the configuration of “<name>-analytics” with attached parameter group “default-redshift-1.0” with a screenshot of parameter group configuration.

```

--profile [REDACTED] east-region redshift describe-cluster-parameter-groups
{
  "ParameterGroups": [
    {
      "ParameterGroupFamily": "redshift-1.0",
      "Tags": [],
      "ParameterGroupName": "default.redshift-1.0",
      "Description": "Default parameter group for redshift-1.0"
    }
  ]
}

--profile [REDACTED] east-region redshift describe-cluster-parameters --parameter-group-name default.redshift-1.0
{
  "Parameters": [
    {
      "Name": "enable_user_activity_logging",
      "Value": "false",
      "Description": "parameter for audit logging purpose",
      "DataType": "boolean",
      "IsModifiable": true,
      "AllowedValues": "true,false",
      "Source": "engine-default",
      "ParameterType": "static",
      "ApplyType": "static"
    }
  ]
}

```

User activity logging disabled

## 4.4 S3

### 4.4.1 World Writable Buckets – All Users

After reviewing S3 bucket ACLs, we found some of the buckets are world-writable for all users, i.e., anonymous users. This configuration will provide FULL\_CONTROL to everyone, i.e., an anonymous user to view, upload, modify, delete S3 objects, view and delete access permission, which can lead to data loss and unexpected charges on AWS bill.

An attacker can upload controversial data on the S3 bucket, which can harm the reputation of the organization, or can delete important data, which will make the IoT ecosystem non-functioning for some period.

Following are the affected buckets:

1. <name1>world
2. <name2>-device-logs-public

Following is a snapshot of ACL of “<name2>-device-logs-public”

```
aws --profile [redacted] s3api get-bucket-acl --bucket [redacted] --profile device-logs-public
```

```
{
  "Grantee": {
    "Type": "Group",
    "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
  },
  "Permission": "WRITE"
}
```

**Recommendation** – It is recommended to implement S3 ACL as restrictive as possible, and only authorized users should have access to the S3 buckets.

**Reference** - <http://docs.aws.amazon.com/AmazonS3/latest/dev/s3-access-control.html>

#### 4.4.2 World Readable and Writable Buckets – Authenticated Users

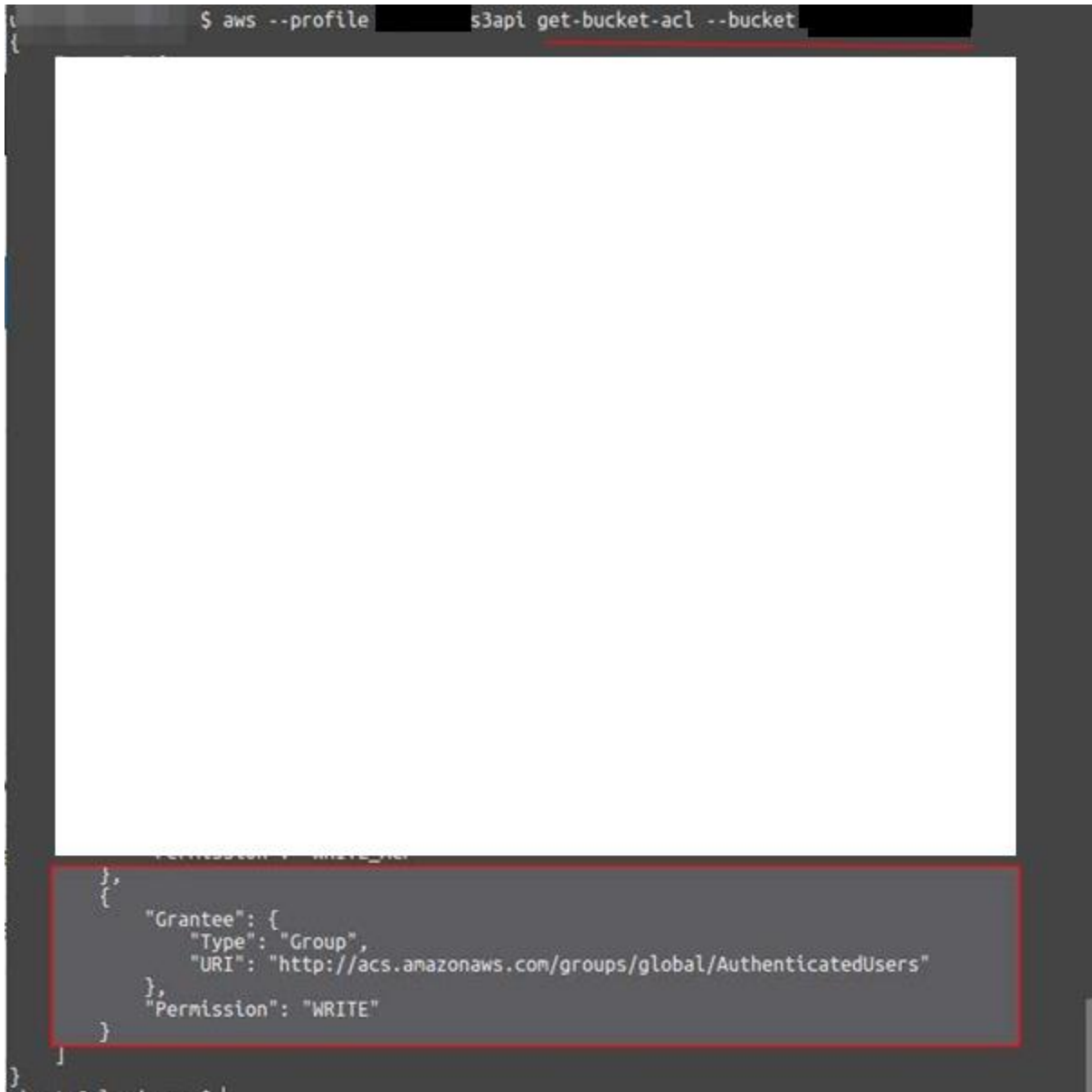
After reviewing S3 bucket ACLs, we found some of the buckets are world-readable and writable for authenticated users. This configuration will provide FULL\_CONTROL to the authenticated user to view, upload, modify, delete S3 objects, view and delete access permission, which can lead to data loss and unexpected charges on the AWS bill.

An authenticated AWS user (disgruntled user) can upload controversial data on the S3 bucket, which can harm the reputation of the organization or can delete important data, which will make the IoT ecosystem non-functioning for some period.

Following are the affected buckets:

1. <name1> (world readable)
2. <name> (world writable)

Following is the screenshot of world writable S3 bucket "<name2".



```
$ aws --profile [redacted] s3api get-bucket-acl --bucket [redacted]
{
  "Grantee": {
    "Type": "Group",
    "URI": "http://acs.amazonaws.com/groups/global/AuthenticatedUsers"
  },
  "Permission": "WRITE"
}
```

**Recommendation** – It is recommended to implement S3 ACL as restrictive as possible, and only authorized users should have access to the S3 bucket objects.

**Reference** - <http://docs.aws.amazon.com/AmazonS3/latest/dev/s3-access-control.html>

#### 4.4.3 World Readable Buckets – All Users

After reviewing S3 bucket ACLs, we found some of the buckets are world-readable for all users, i.e., anonymous users. This configuration will provide everyone, i.e., an anonymous user to list the objects of an S3 bucket.

Anyone on the internet can access and download the files present on S3 buckets and can modify them to install backdoors and redistribute on behalf of the service provider.

Following is the affected buckets:

1. <name1>

Following is the ACL of the bucket "<name1>."

```
aws --profile [REDACTED] s3api get-bucket-acl --bucket [REDACTED]
{
  "Grantee": {
    "Type": "Group",
    "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
  },
  "Permission": "READ"
}
```

**Recommendation** – It is recommended to implement S3 ACL as restrictive as possible, and only authorized users should have access to the S3 bucket objects.

**Reference** - <http://docs.aws.amazon.com/AmazonS3/latest/dev/s3-access-control.html>



**Recommendation** – Enable "Logging" property of the S3 bucket containing business-critical files. By setting this property, AWS admin can get details of the operation performed on the files of an S3 bucket, and it will be helpful for security audits.

**Reference** – <http://docs.aws.amazon.com/AmazonS3/latest/dev/ServerLogs.html>

#### 4.4.5 Versioning

After reviewing the properties of S3 buckets, we found that the "versioning" properties of the implemented bucket are disabled. AWS will not maintain different versions of a particular file thus will not be able to preserve and recover overwritten and deleted S3 objects.

Following image shows the bucket with disabled versioning property.

[illegible]





**Recommendation** – Enable “MFADelete” property of S3 bucket containing business critical files. By setting this property, the S3 bucket will prevent the deletion of a file by an unauthenticated user.

**Reference** – <http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingMFADelete.html>

## 4.5 EC2

### 4.5.1 Ports open to all

After reviewing the security group, it was found that critical TCP ports 22 (SSH) and 27017 (MongoDB) and other TCP and UDP ports are open to everyone means everyone on the internet can carry out service-based attacks.

An attacker can launch an SSH-based attack to gain access to an EC2 instance and can use a compromised EC2 instance for attacking other AWS services or systems over the internet.

Following security groups are misconfigured.

1. sg-d5a58bb1
2. sg-15c3247c
3. sg-39ef0550
4. sg-fb73b292
5. sg-1a29d77c
6. sg-950594f0
7. sg-d40594b1
8. sg-d27f81b4
9. sg-863ac4e0
10. sg-e0c93786
11. sg-d5a58bb1
12. sg-0a530571
13. sg-0d660969
14. sg-226a9346
15. sg-6314f507
16. sg-9d47f6f9
17. sg-5963d53e
18. sg-70cd4d17
19. sg-a8e2bacc
20. sg-c0db6ba7
21. sg-6c752109
22. sg-d4c268b0
23. sg-dbf0a1be
24. sg-9c1070f8
25. sg-873dede3
26. sg-96b923f2

Following is a screenshot of security group “sg-d5a58bb1”

AWS Security Configuration Review Report	Document Level: Confidential
Document Version: 1.0	Page 25 of 28

```

$ aws --profile [redacted] --region [redacted] ec2 describe-security-groups --filter Name=group-id,Values=sg-d5a58bb1
{
  "SecurityGroups": [
    {
      "PrefixListIds": [],
      "FromPort": 22,
      "IpRanges": [
        {
          "CidrIp": "0.0.0.0/0"
        }
      ],
      "ToPort": 22,
      "IpProtocol": "tcp",
      "UserIdGroupPairs": [],
      "Ipv6Ranges": []
    },
    {
      "PrefixListIds": [],
      "FromPort": 27017,
      "IpRanges": [
        {
          "CidrIp": "0.0.0.0/0"
        }
      ],
      "ToPort": 27017,
      "IpProtocol": "tcp",
      "UserIdGroupPairs": [],
      "Ipv6Ranges": []
    }
  ],
  "GroupName": "[redacted]",
  "VpcId": "[redacted]",
  "OwnerId": "[redacted]",
  "GroupId": "[redacted]"
}

```

**Recommendation** – Restrict access to TCP port 22(SSH) and 27017 (MongoDB) to required entities. Minimize the number of discrete security groups to decrease the risk of misconfiguration leading.

**Reference** – <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html>

#### 4.5.2 Default Security Groups

After reviewing security groups, we found that 6 default security groups are in use. If the default security group allows unrestricted access and it increases surface areas for malicious activity.

The default security groups are configured for general access control. The use of these groups can unintentionally allow an unauthorized user to communicate with the EC2 instance.

Following are default security groups in use

1. sg-4d8d2028
2. sg-93dd3cf7
3. sg-9858e0fc

4. sg-8ce4dcf6
5. sg-b5ec1ddc
6. sg-db3dedbf
7. sg-1ea58378
8. sg-a3f3a6c6
9. sg-6c752109
10. sg-37ee744e
11. sg-bd5103d9

Following is a screenshot of default security groups in use.

us-west-2 security groups:

```
aws --profile [redacted] describe-security-groups --filter Name=group-name,Values=default --query "SecurityGroups[*].(Name:GroupId)"
[
  {
    "Name": "sg-1ea58378"
  },
  {
    "Name": "sg-682b4b0c"
  },
  {
    "Name": "sg-6c752109"
  },
  {
    "Name": "sg-a3f3a6c6"
  },
  {
    "Name": "sg-db3dedbf"
  }
]
```

**Recommendation** – Use user-defined security group with a restriction on inbound connection and should conform to least privilege rule of AWS security.

**Reference** –

[http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC\\_SecurityGroups.html](http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_SecurityGroups.html)