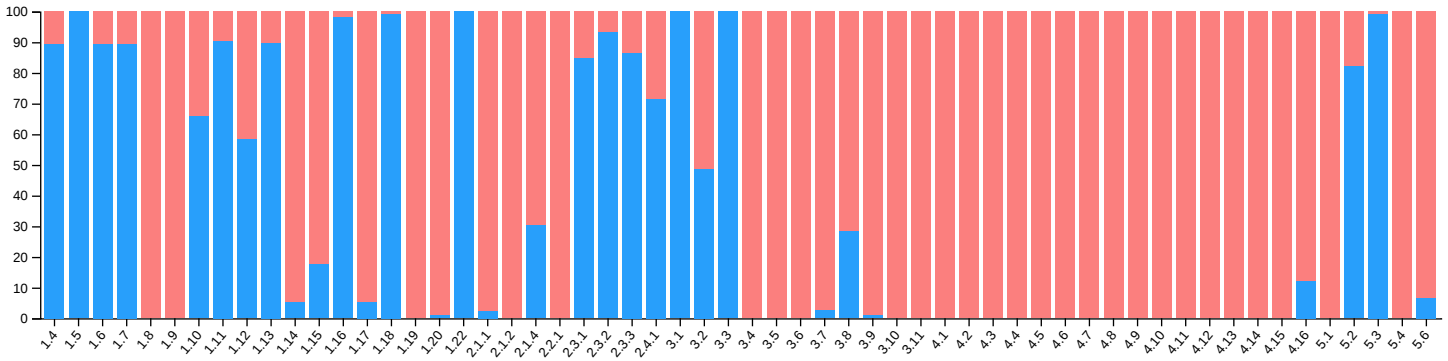
Report Name - CIS - Tevo1ALL- AWS | Date generated(UTC): 27/6/2025 0:1:11

Total Accounts Integrated	Resources Scanned	Failed	Passed
19	22939	10551	12388



Compliance Summary

Section	Title	Failed	Passed
1.4	Ensure no 'root' user account access key exists	2	17
1.5	Ensure MFA is enabled for the 'root' user account	0	19
1.6	Ensure hardware MFA is enabled for the 'root' user account	2	17
1.7	Eliminate use of the 'root' user for administrative and daily tasks	2	17
1.8	Ensure IAM password policy requires minimum length of 14 or greater	19	0
1.9	Ensure IAM password policy prevents password reuse	19	0
1.10	Ensure multi-factor authentication (MFA) is enabled for all IAM users that have a console password	18	35
1.11	Do not setup access keys during initial user setup for all IAM users that have a console password	17	159
1.12	Ensure credentials unused for 45 days or greater are disabled	65	92
1.13	Ensure there is only one active access key available for any single IAM user	18	158
1.14	Ensure access keys are rotated every 90 days or less	121	7
1.15	Ensure IAM Users Receive Permissions Only Through Groups	129	28
1.16	Ensure IAM policies that allow full "*:*" administrative privileges are not attached	86	5111
1.17	Ensure a support role has been created to manage incidents with AWS Support	18	1
1.18	Ensure IAM instance roles are used for AWS resource access from instances	31	3674
1.19	Ensure that all the expired SSL/TLS certificates stored in AWS IAM are removed	2	0
1.20	Ensure that IAM Access analyzer is enabled for all regions	320	3
1.22	Ensure access to AWSCloudShellFullAccess is restricted	0	19
2.1.1	Ensure S3 Bucket Policy is set to deny HTTP requests	689	17
2.1.2	Ensure MFA Delete is enabled on S3 buckets	706	0
2.1.4	Ensure that S3 Buckets are configured with 'Block public access (bucket settings)'	492	214
2.2.1	Ensure EBS Volume Encryption is Enabled in all Regions	323	0
2.3.1	Ensure that encryption-at-rest is enabled for RDS Instances	9	50

Section	Title	Failed	Passed
2.3.2	Ensure Auto Minor Version Upgrade feature is Enabled for RDS Instances	4	55
2.3.3	Ensure that public access is not given to RDS Instance	8	51
2.4.1	Ensure that encryption is enabled for EFS file systems	2	5
3.1	Ensure CloudTrail is enabled in all regions	0	19
3.2	Ensure CloudTrail log file validation is enabled	20	19
3.3	Ensure the S3 bucket used to store CloudTrail logs is not publicly accessible	0	2
3.4	Ensure CloudTrail trails are integrated with CloudWatch Logs	39	0
3.5	Ensure AWS Config is enabled in all regions	323	0
3.6	Ensure S3 bucket access logging is enabled on the CloudTrail S3 bucket	39	0
3.7	Ensure CloudTrail logs are encrypted at rest using KMS CMKs	38	1
3.8	Ensure rotation for customer created symmetric CMKs is enabled	25	10
3.9	Ensure VPC flow logging is enabled in all VPCs	422	5
3.10	Ensure that Object-level logging for write events is enabled for S3 bucket	706	0
3.11	Ensure that Object-level logging for read events is enabled for S3 bucket	706	0
4.1	Ensure unauthorized API calls are monitored	19	0
4.2	Ensure management console sign-in without MFA is monitored	19	0
4.3	Ensure usage of 'root' account is monitored	19	0
4.4	Ensure IAM policy changes are monitored	19	0
4.5	Ensure CloudTrail configuration changes are monitored	19	0
4.6	Ensure AWS Management Console authentication failures are monitored	19	0
4.7	Ensure disabling or scheduled deletion of customer created CMKs is monitored	19	0
4.8	Ensure S3 bucket policy changes are monitored	19	0
4.9	Ensure AWS Config configuration changes are monitored	19	0
4.10	Ensure security group changes are monitored	19	0
4.11	Ensure Network Access Control Lists (NACL) changes are monitored	19	0
4.12	Ensure changes to network gateways are monitored	19	0
4.13	Ensure route table changes are monitored	19	0
4.14	Ensure VPC changes are monitored	19	0
4.15	Ensure AWS Organizations changes are monitored	19	0
4.16	Ensure AWS Security Hub is enabled	284	39
5.1	Ensure no Network ACLs allow ingress from 0.0.0.0/0 to remote server administration ports	445	0
5.2	Ensure no security groups allow ingress from 0.0.0.0/0 to remote server administration ports	223	1043
5.3	Ensure no security groups allow ingress from ::/0 to remote server administration ports	10	1256
5.4	Ensure the default security group of every VPC restricts all traffic	423	1
5.6	Ensure that EC2 Metadata Service only allows IMDSv2	3461	244

Control: Ensure no 'root' user account access key exists

Sections: 1.4 2 17

Description: The 'root' user account is the most privileged user in an AWS account. AWS Access Keys provide programmatic access to a given AWS account. It is recommended that all access keys associated with the 'root' user account be deleted.

Remediation Steps:
Perform the following to delete active 'root' user access keys.

From Console:

1. Sign in to the AWS Management Console as 'root' and open the IAM console at https://console.aws.amazon.com/iam/.
2. Click on <root_account> at the top right and select **My Security Credentials** from the drop down list.
3. On the pop out screen Click on **Continue to Security Credentials**.
4. Click on **Access Keys** (Access Key ID and Secret Access Key).
5. Under the **Status** column (if there are any Keys which are active).
6. Click **Delete** (Note: Deleted keys cannot be recovered).

Note: While a key can be made inactive, this inactive key will still show up in the CLI command from the audit procedure, and may lead to a key being falsely flagged as being non-compliant.

Control: Ensure hardware MFA is enabled for the 'root' user account

Sections: 1.6 2 17

Description: The 'root' user account is the most privileged user in an AWS account. MFA adds an extra layer of protection on top of a user name and password. With MFA enabled, when a user signs in to an AWS website, they will be prompted for their user name and password as well as for an authentication code from their AWS MFA device. For Level 2, it is recommended that the 'root' user account be protected with a hardware MFA.

Remediation Steps:
Perform the following to establish a hardware MFA for the 'root' user account:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/. Note: to manage MFA devices for the AWS 'root' user account, you must use your 'root' account credentials to sign in to AWS. You cannot manage MFA devices for the 'root' account using other credentials.
2. Choose **Dashboard** , and under **Security Status** , expand **Activate MFA** on your root account.
3. Choose **Activate MFA**
4. In the wizard, choose **A hardware MFA** device and then choose **Next Step** .
5. In the Serial Number box, enter the serial number that is found on the back of the MFA device.
6. In the **Authentication Code 1** box, enter the six-digit number displayed by the MFA device. You might need to press the button on the front of the device to display the number.
7. Wait 30 seconds while the device refreshes the code, and then enter the next six-digit number into the Authentication Code 2 box. You might need to press the button on the front of the device again to display the second number.
8. Choose **Next Step** . The MFA device is now associated with the AWS account. The next time you use your AWS account credentials to sign in, you must type a code from the hardware MFA device.

Remediation for this recommendation is not available through AWS CLI.

Control: Eliminate use of the 'root' user for administrative and daily tasks

Sections: 1.7 2 17

Description: With the creation of an AWS account, a 'root user' is created that cannot be disabled or deleted. That user has unrestricted access to and control over all resources in the AWS account. It is highly recommended that the use of this account be avoided for everyday tasks.

Remediation Steps:
If you find that the 'root' user account is being used for daily activity to include administrative tasks that do not require the 'root' user:

1. Change the 'root' user password.
2. Deactivate or delete any access keys associate with the 'root' user.

**Remember, anyone who has 'root' user credentials for your AWS account has unrestricted access to and control of all the resources in your account, including billing information.

Control: Ensure IAM password policy requires minimum length of 14 or greater

Sections: 1.8 19 0

Description: Password policies are, in part, used to enforce password complexity requirements. IAM password policies can be used to ensure password are at least a given length. It is recommended that the password policy require a minimum password length 14.

Remediation Steps:
Perform the following to set the password policy as prescribed:

From Console:

1. Login to AWS Console (with appropriate permissions to View Identity Access Management Account Settings)
2. Go to IAM Service on the AWS Console
3. Click on Account Settings on the Left Pane
4. Set "Minimum password length" to 14 or greater.
5. Click "Apply password policy"

From Command Line:

```
aws iam update-account-password-policy --minimum-password-length 14
```

Note: All commands starting with "aws iam update-account-password-policy" can be combined into a single command.

Control: Ensure IAM password policy prevents password reuse

Sections: 1.9 19 0

Description: IAM password policies can prevent the reuse of a given password by the same user. It is recommended that the password policy prevent the reuse of passwords.

Remediation Steps:
Perform the following to set the password policy as prescribed:

From Console:

1. Login to AWS Console (with appropriate permissions to View Identity Access Management Account Settings)
2. Go to IAM Service on the AWS Console
3. Click on Account Settings on the Left Pane
4. Check "Prevent password reuse"
5. Set "Number of passwords to remember" is set to 24

From Command Line:

```
aws iam update-account-password-policy --password-reuse-prevention 24
```

Note: All commands starting with "aws iam update-account-password-policy" can be combined into a single command.

Control: Ensure multi-factor authentication (MFA) is enabled for all IAM users that have a console password

Sections: 1.10 18 35

Description: Multi-Factor Authentication (MFA) adds an extra layer of authentication assurance beyond traditional credentials. With MFA enabled, when a user signs in to the AWS Console, they will be prompted for their user name and password as well as for an authentication code from their physical or virtual MFA token. It is recommended that MFA be enabled for all accounts that have a console password.

Remediation Steps:
Perform the following to enable MFA:

From Console:

1. Sign in to the AWS Management Console and open the IAM console at 'https://console.aws.amazon.com/iam/'
2. In the left pane, select **Users**.
3. In the **User Name** list, choose the name of the intended MFA user.
4. Choose the **Security Credentials** tab, and then choose **Manage MFA Device**.
5. In the **Manage MFA Device wizard**, choose **Virtual MFA device**, and then choose **Continue**.

   IAM generates and displays configuration information for the virtual MFA device, including a QR code graphic. The graphic is a representation of the 'secret configuration key' that is available for manual entry on devices that do not support QR codes.
6. Open your virtual MFA application. (For a list of apps that you can use for hosting virtual MFA devices, see Virtual MFA Applications at https://aws.amazon.com/iam/details/mfa/#Virtual_MFA_Applications). If the virtual MFA application supports multiple accounts (multiple virtual MFA devices), choose the option to create a new account (a new virtual MFA device).
7. Determine whether the MFA app supports QR codes, and then do one of the Following:
   - Use the app to scan the QR code. For example, you might choose the camera icon or choose an option similar to Scan code, and then use the device's camera to scan the code.
   - In the Manage MFA Device wizard, choose Show secret key for manual configuration, and then type the secret configuration key into your MFA application.

   When you are finished, the virtual MFA device starts generating one-time passwords.
8. In the **Manage MFA Device wizard**, in the **MFA Code 1 box**, type the **one-time password** that currently appears in the virtual MFA device. Wait up to 30 seconds for the device to generate a new one-time password. Then type the second onetime password into the **MFA Code 2 box**.
9. Click **Assign MFA**.

Control: Do not setup access keys during initial user setup for all IAM users that have a console password

Sections: 1.11 17 159

Description: AWS console defaults to no check boxes selected when creating a new IAM user. When creating the IAM User credentials you have to determine what type of access they require. Programmatic access: The IAM user might need to make API calls, use the AWS CLI, or use the Tools for Windows PowerShell. In that case, create an access key (access key ID and a secret access key) for that user. AWS Management Console access: If the user needs to access the AWS Management Console, create a password for the user.

Remediation Steps:
Perform the following to delete access keys that do not pass the audit:

From Console:

1. Login to the AWS Management Console:
2. Click **Services**
3. Click **IAM**
4. Click on **Users**
5. Click on **Security Credentials**
6. As an **Administrator**

- Click on the X (Delete) for keys that were created at the same time as the user profile but have not been used.

7. As an IAM User

- Click on the X (Delete) for keys that were created at the same time as the user profile but have not been used.

From Command Line:

```
aws iam delete-access-key --access-key-id <access-key-id-listed> --user-name <users-name>
```

Control: Ensure credentials unused for 45 days or greater are disabled

Sections: 1.12 65 92

Description: AWS IAM users can access AWS resources using different types of credentials, such as passwords or access keys. It is recommended that all credentials that have been unused in 45 or greater days be deactivated or removed.

Remediation Steps:
From Console:

Perform the following to manage Unused Password (IAM user console access)

1. Login to the AWS Management Console:
2. Click **Services**
3. Click **IAM**
4. Click on **Users**
5. Click on **Security Credentials**
6. Select user whose Console last sign-in is greater than **45 days**
7. Click **Security credentials**
8. In section Sign-in credentials, **Console password** click **Manage**
9. Under **Console Access** select **Disable**
10. Click **Apply**

Perform the following to deactivate Access Keys:

1. Login to the AWS Management Console:
2. Click **Services**
3. Click **IAM**
4. Click on **Users**
5. Click on **Security Credentials**
6. Select any access keys that are over 45 days old and that have been used and

- Click on **Make Inactive**

7. Select any access keys that are over 45 days old and that have not been used And

- Click the X to **Delete**

Control: Ensure there is only one active access key available for any single IAM user

Sections: 1.13 18 158

Description: Access keys are long-term credentials for an IAM user or the AWS account 'root' user. You can use access keys to sign programmatic requests to the AWS CLI or AWS API (directly or using the AWS SDK)

Remediation Steps:
From Console:

1. Sign in to the AWS Management Console and navigate to IAM dashboard at https://console.aws.amazon.com/iam/.
2. In the left navigation panel, choose **Users**.
3. Click on the **IAM user name** that you want to **examine**.
4. On the IAM user configuration page, select **Security Credentials tab**.
5. In **Access Keys** section, choose one access key that is less than 90 days old. This should be the only active key used by this IAM user to access AWS resources programmatically. Test your application(s) to make sure that the chosen access key is working.
6. In the same Access Keys section, identify your non-operational access keys (other than the chosen one) and deactivate it by clicking the **Make Inactive** link.
7. If you receive the **Change Key Status** confirmation box, click **Deactivate** to switch off the selected key.
8. Repeat steps no. 3 and 7 for each IAM user in your AWS account.

From Command Line:

1. Using the IAM user and access key information provided in the Audit CLI, choose one access key that is less than 90 days old. This should be the only active key used by this IAM user to access AWS resources programmatically. Test your application(s) to make sure that the chosen access key is working.
2. Run the **update-access-key** command below using the IAM user name and the non-operational access key IDs to deactivate the unnecessary key(s). Refer to the Audit section to identify the unnecessary access key ID for the selected IAMUser

   Note - the command does not return any output:

   ```
   aws iam update-access-key --access-key-id <access-key-id> --status Inactive --user-name <user-name>
   ```
3. To confirm that the selected access key pair has been successfully **deactivated** run the **list-access-keys** audit command again for that IAM User:

   ```
   aws iam list-access-keys --user-name <user-name>
   ```

- The command output should expose the metadata for each access key associated with the IAM user. If the non-operational key pair(s) **Status** is set to **Inactive**, the key has been successfully deactivated and the IAM user access configuration adheres now to this recommendation.

4. Repeat steps no. 1 and 3 for each IAM user in your AWS account.

Control: Ensure access keys are rotated every 90 days or less

Sections: 1.14 121 7

Description: Access keys consist of an access key ID and secret access key, which are used to sign programmatic requests that you make to AWS. AWS users need their own access keys to make programmatic calls to AWS from the AWS Command Line Interface (AWS CLI), Tools for Windows PowerShell, the AWS SDKs, or direct HTTP calls using the APIs for individual AWS services. It is recommended that all access keys be regularly rotated.

Remediation Steps:
Perform the following to rotate access keys:

From Console:

1. Go to Management Console (https://console.aws.amazon.com/iam)
2. Click on **Users**
3. Click on **Security Credentials**
4. As an **Administrator**
   - Click on **Make Inactive** for keys that have not been rotated in 90 Days
5. As an **IAM User**
   - Click on Make Inactive or Delete for keys which have not been rotated or used in 90 Days
6. Click on **Create Access Key**
7. Update programmatic call with new **Access Key credentials**

From Command Line:

1. While the first access key is still active, create a second access key, which is active by default. Run the following command:

   ```
   aws iam create-access-key
   ```

   At this point, the user has two active access keys.
2. Update all applications and tools to use the new access key.
3. Determine whether the first access key is still in use by using this command:

   ```
   aws iam get-access-key-last-used
   ```
4. One approach is to wait several days and then check the old access key for any use before proceeding.

   Even if step Step 3 indicates no use of the old key, it is recommended that you do not immediately delete the first access key. Instead, change the state of the first access key to Inactive using this command:

   ```
   aws iam update-access-key
   ```
5. Use only the new access key to confirm that your applications are working. Any applications and tools that still use the original access key will stop working at this point because they no longer have access to AWS resources. If you find such an application or tool, you can switch its state back to Active to reenable the first access key. Then return to step Step 2 and update this application to use the new key.
6. After you wait some period of time to ensure that all applications and tools have been updated, you can delete the first access key with this command:

   ```
   aws iam delete-access-key
   ```

Control: Ensure IAM Users Receive Permissions Only Through Groups

Sections: 1.15 129 28

Description: IAM users are granted access to services, functions, and data through IAM policies. There are four ways to define policies for a user: 1) Edit the user policy directly, aka an inline, or user, policy; 2) attach a policy directly to a user; 3) add the user to an IAM group that has an attached policy; 4) add the user to an IAM group that has an inline policy. Only the third implementation is recommended.

Remediation Steps:
Perform the following to create an IAM group and assign a policy to it:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, click **Groups** and then click **Create New Group** .
3. In the **Group Name box**, type the name of the group and then click Next Step .
4. In the list of policies, select the check box for each policy that you want to apply to all members of the group. Then click **Next Step** .
5. Click **Create Group**

Perform the following to add a user to a given group:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, click **Groups**
3. Select the group to add a user to
4. Click **Add Users To Group**
5. Select the **users** to be added to the group
6. Click **Add Users**

Perform the following to remove a direct association between a user and policy:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, click on **Users**
3. For each user:
    - Select the **user**
    - Click on the **Permissions tab**
    - Expand **Permissions policies**
    - Click X for each policy; then click **Detach** or **Remove** (depending on policy type)

Control: Ensure IAM policies that allow full "*:*" administrative privileges are not attached

Sections: 1.16 86 5111

Description: IAM policies are the means by which privileges are granted to users, groups, or roles. It is recommended and considered a standard security advice to grant least privilege -that is, granting only the permissions required to perform a task. Determine what users need to do and then craft policies for them that let the users perform only those tasks, instead of allowing full administrative privileges.

Remediation Steps:
From Console:

Perform the following to detach the policy that has full administrative privileges:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, click Policies and then search for the policy name found in the audit step.
3. Select the **policy** that needs to be **deleted**.
4. In the policy action menu, select first **Detach**
5. Select all **Users, Groups**, Roles that have this policy attached
6. Click **Detach Policy**
7. In the policy action menu, select **Detach**

From Command Line:

Perform the following to detach the policy that has full administrative privileges as found in the audit step:

1. Lists all IAM users, groups, and roles that the specified managed policy is attached to.

   ```
   aws iam list-entities-for-policy --policy-arn <policy_arn>
   ```
2. Detach the policy from all **IAM Users**:

   ```
   aws iam detach-user-policy --user-name <iam_user> --policy-arn <policy_arn>
   ```
3. Detach the policy from all **IAM Groups**:

   ```
   aws iam detach-group-policy --group-name <iam_group> --policy-arn <policy_arn>
   ```
4. Detach the policy from all **IAM Roles**:

   ```
   aws iam detach-role-policy --role-name <iam_role> --policy-arn <policy_arn>
   ```

Control: Ensure a support role has been created to manage incidents with AWS Support

Sections: 1.17 18 1

Description: AWS provides a support center that can be used for incident notification and response, as well as technical support and customer services. Create an IAM Role, with the appropriate policy assigned, to allow authorized users to manage incidents with AWS Support.

Remediation Steps:
From Command Line:

1. Create an IAM role for managing incidents with AWS:
   - Create a trust relationship policy document that allows <iam_user> to manage AWS incidents, and save it locally as /tmp/TrustPolicy.json:

   ```
   { "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "AWS": "<iam_user>" }, "Action": "sts:AssumeRole"
   } ] }
   ```
2. Create the IAM role using the above trust policy:

   ```
   aws iam create-role --role-name <aws_support_iam_role> --assume-role-policydocument file:///tmp/TrustPolicy.json
   ```
3. Attach 'AWSSupportAccess' managed policy to the created IAM role:

   ```
   aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/AWSSupportAccess --role-name <aws_support_iam_role>
   ```

Control: Ensure IAM instance roles are used for AWS resource access from instances

Sections: 1.18 31 3674

Description: 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 Access" means accessing the APIs of AWS in order to access AWS resources or manage AWS account resources.

Remediation Steps:
From Console:

1. Sign in to the AWS Management Console and navigate to EC2 dashboard at https://console.aws.amazon.com/ec2/.
2. In the left navigation panel, choose Instances.
3. Select the EC2 instance you want to modify.
4. Click Actions.
5. Click Security.
6. Click Modify IAM role.
7. Click Create new IAM role if a new IAM role is required.
8. Select the IAM role you want to attach to your instance in the IAM role dropdown.
9. Click Update IAM role. 10.Repeat steps 3 to 9 for each EC2 instance in your AWS account that requires an IAM role to be attached.

From Command Line:

1. Run the describe-instances command to list all EC2 instance IDs, available in the selected AWS region:
   ```
   aws ec2 describe-instances --region <region-name> --query 'Reservations[*].Instances[*].InstanceId'
   ```
2. Run the associate-iam-instance-profile command to attach an instance profile (which is attached to an IAM role) to the EC2 instance:
   ```
   aws ec2 associate-iam-instance-profile --region <region-name> --instance-id <Instance-ID> --iam-instance-profile Name="Instance-Profile-Name"
   ```
3. Run the describe-instances command again for the recently modified EC2 instance. The command output should return the instance profile ARN and ID:
   ```
   aws ec2 describe-instances --region <region-name> --instance-id <Instance-ID> --query 'Reservations[*].Instances[*].IamInstanceProfile'
   ```
4. Repeat steps 1 to 3 for each EC2 instance in your AWS account that requires an IAM role to be attached.

Control: Ensure that all the expired SSL/TLS certificates stored in AWS IAM are removed

Sections: 1.19 2 0

Description: To enable HTTPS connections to your website or application in AWS, you need an SSL/TLS server certificate. You can use ACM or IAM to store and deploy server certificates. Use IAM as a certificate manager only when you must support HTTPS connections in a region that is not supported by ACM. IAM securely encrypts your private keys and stores the encrypted version in IAM SSL certificate storage. IAM supports deploying server certificates in all regions, but you must obtain your certificate from an external provider for use with AWS. You cannot upload an ACM certificate to IAM. Additionally, you cannot manage your certificates from the IAM Console.

Remediation Steps:
From Console: Removing expired certificates via AWS Management Console is not currently supported. To delete SSL/TLS certificates stored in IAM via the AWS API use the Command Line Interface (CLI).

From Command Line:

1. To delete Expired Certificate run following command by replacing <CERTIFICATE_NAME> with the name of the certificate to delete:
   ```
   aws iam delete-server-certificate --server-certificate-name <CERTIFICATE_NAME>
   ```
   When the preceding command is successful, it does not return any output.

Control: Ensure that IAM Access analyzer is enabled for all regions

Sections: 1.20 320 3

Description: Enable IAM Access analyzer for IAM policies about all resources in each active AWS region. IAM Access Analyzer is a technology introduced at AWS reinvent 2019. After the Analyzer is enabled in IAM, scan results are displayed on the console showing the accessible resources. Scans show resources that other accounts and federated users can access, such as KMS keys and IAM roles. So the results allow you to determine if an unintended user is allowed, making it easier for administrators to monitor least privileges access. Access Analyzer analyzes only policies that are applied to resources in the same AWS Region.

Remediation Steps:

From Console:

Perform the following to enable IAM Access analyzer for IAM policies:

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose Access analyzer.
3. Choose Create analyzer.
4. On the Create analyzer page, confirm that the Region displayed is the Region where you want to enable Access Analyzer.
5. Enter a name for the analyzer. Optional as it will generate a name for you automatically.
6. Add any tags that you want to apply to the analyzer. Optional.
7. Choose Create Analyzer.
8. Repeat these step for each active region

From Command Line:

1. Run the following command:

   ```
   aws accessanalyzer create-analyzer --analyzer-name <NAME> --type <ACCOUNT|ORGANIZATION>
   ```

   Repeat this command above for each active region.

   Note: The IAM Access Analyzer is successfully configured only when the account you use has the necessary permissions.

Control: Ensure S3 Bucket Policy is set to deny HTTP requests

Sections: 2.1.1 689 17

Description: At the Amazon S3 bucket level, you can configure permissions through a bucket policy making the objects accessible only through HTTPS.

Remediation Steps:
From Console:

1. Login to AWS Management Console and open the Amazon S3 console using https://console.aws.amazon.com/s3/
2. Select the Check box next to the Bucket.
3. Click on 'Permissions'.
4. Click 'Bucket Policy'
5. Add this to the existing policy filling in the required information

```
{ "Sid": <optional>", "Effect": "Deny", "Principal": "*", "Action": "s3:*", "Resource": "arn:aws:s3:::<bucket\_name>/*",
"Condition": { "Bool": { “aws:SecureTransport": "false" } } }
```

6. Save
7. Repeat for all the buckets in your AWS account that contain sensitive data.

From Console using AWS Policy Generator:

1. Repeat steps 1-4 above.
2. Click on Policy Generator at the bottom of the Bucket Policy Editor
3. Select Policy Type S3 Bucket Policy
4. Add Statements

- Effect = Deny
- Principal = *
- AWS Service = Amazon S3
- Actions = *
- Amazon Resource Name = <ARN of the S3 Bucket>

5. Generate Policy
6. Copy the text and add it to the Bucket Policy.

From Command Line:

1. Export the bucket policy to a json file.

```
aws s3api get-bucket-policy --bucket <bucket\_name> --query Policy --output text > policy.json
```

2. Modify the policy.json file by adding in this statement:

```
{ "Sid": <optional>", "Effect": "Deny", "Principal": "*", "Action": "s3:*", “Resource": "arn:aws:s3:::<bucket\_name>/*",
"Condition": { "Bool": { "aws:SecureTransport": "false" } } }
```

3. Apply this modified policy back to the S3 bucket:

```
aws s3api put-bucket-policy --bucket <bucket\_name> --policy file://policy.json
```

Control: Ensure MFA Delete is enabled on S3 buckets

Sections: 2.1.2 706 0

Description: Once MFA Delete is enabled on your sensitive and classified S3 bucket it requires the user to have two forms of authentication.

Remediation Steps:
Perform the steps below to enable MFA delete on an S3 bucket.

Note:

- You cannot enable MFA Delete using the AWS Management Console. You must use the AWS CLI or API.
- You must use your 'root' account to enable MFA Delete on S3 buckets.

From Command Line:

1. Run the s3api put-bucket-versioning command

```
aws s3api put-bucket-versioning --profile my-root-profile –bucket Bucket_Name --versioning-configuration
Status=Enabled,MFADelete=Enabled --mfa “arn:aws:iam::aws_account_id:mfa/root-account-mfa-device passcode”
```

Control: Ensure that S3 Buckets are configured with 'Block public access (bucket settings)'

Sections: 2.1.4 492 214

Description: Amazon S3 provides Block public access (bucket settings) and Block public access (account settings) to help you manage public access to Amazon S3 resources. By default, S3 buckets and objects are created with public access disabled. However, an IAM principal with sufficient S3 permissions can enable public access at the bucket and/or object level. While enabled, Block public access (bucket settings) prevents an individual bucket, and its contained objects, from becoming publicly accessible. Similarly, Block public access (account settings) prevents all buckets, and contained objects, from becoming publicly accessible across the entire account.

Remediation Steps:
If utilizing Block Public Access (bucket settings)

From Console:

1. Login to AWS Management Console and open the Amazon S3 console using https://console.aws.amazon.com/s3/
2. Select the Check box next to the Bucket.
3. Click on 'Edit public access settings'.
4. Click 'Block all public access'
5. Repeat for all the buckets in your AWS account that contain sensitive data.

From Command Line:

1. List all of the S3 Buckets

```
aws s3 ls
```

2. Set the Block Public Access to true on that bucket

```
aws s3api put-public-access-block --bucket <name-of-bucket> --public-accessblock-configuration
"BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPu
```

blicBuckets=true" ```

If utilizing Block Public Access (account settings)

From Console:

If the output reads true for the separate configuration settings then it is set on the account.

1. Login to AWS Management Console and open the Amazon S3 console using https://console.aws.amazon.com/s3/
2. Choose Block Public Access (account settings)
3. Choose Edit to change the block public access settings for all the buckets in your AWS account
4. Choose the settings you want to change, and then choose Save. For details about each setting, pause on the i icons.
5. When you're asked for confirmation, enter confirm. Then Click Confirm to save your changes.

From Command Line: To set Block Public access settings for this account, run the following command:

```
aws s3control put-public-access-block --public-access-block-configuration BlockPublicAcls=true, IgnorePublicAcls=true,
BlockPublicPolicy=true, RestrictPublicBuckets=true --account-id <value>
```

Control: Ensure EBS Volume Encryption is Enabled in all Regions

Sections: 2.2.1 323 0

Description: Elastic Compute Cloud (EC2) supports encryption at rest when using the Elastic Block Store (EBS) service. While disabled by default, forcing encryption at EBS volume creation is supported.

Remediation Steps:

From Console:

1. Login to AWS Management Console and open the Amazon EC2 console using https://console.aws.amazon.com/ec2/
2. Under Account attributes, click EBS encryption.
3. Click Manage.
4. Click the Enable checkbox.
5. Click Update EBS encryption
6. Repeat for every region requiring the change.

Note: EBS volume encryption is configured per region.

From Command Line:

1. Run

```
aws --region <region> ec2 enable-ebs-encryption-by-default
```

2. Verify that "EbsEncryptionByDefault": true is displayed.
3. Repeat every region requiring the change.

Note: EBS volume encryption is configured per region.

Control: Ensure that encryption-at-rest is enabled for RDS Instances

Sections: 2.3.1 9 50

Description: Amazon RDS encrypted DB instances use the industry standard AES-256 encryption algorithm to encrypt your data on the server that hosts your Amazon RDS DB instances. After your data is encrypted, Amazon RDS handles authentication of access and decryption of your data transparently with a minimal impact on performance.

Remediation Steps:
From Console:

1. Login to the AWS Management Console and open the RDS dashboard at https://console.aws.amazon.com/rds/.
2. In the left navigation panel, click on Databases
3. Select the Database instance that needs to be encrypted.
4. Click on Actions button placed at the top right and select Take Snapshot.
5. On the Take Snapshot page, enter a database name of which you want to take a snapshot in the Snapshot Name field and click on Take Snapshot.
6. Select the newly created snapshot and click on the Action button placed at the top right and select Copy snapshot from the Action menu.
7. On the Make Copy of DB Snapshot page, perform the following:
   - In the New DB Snapshot Identifier field, Enter a name for the new snapshot.
   - Check Copy Tags, New snapshot must have the same tags as the source snapshot.
   - Select Yes from the Enable Encryption dropdown list to enable encryption, You can choose to use the AWS default encryption key or custom key from Master Key dropdown list.
8. Click Copy Snapshot to create an encrypted copy of the selected instance snapshot.
9. Select the new Snapshot Encrypted Copy and click on the Action button placed at the top right and select Restore Snapshot button from the Action menu, This will restore the encrypted snapshot to a new database instance. 10.On the Restore DB Instance page, enter a unique name for the new database instance in the DB Instance Identifier field. 11.Review the instance configuration details and click Restore DB Instance. 12.As the new instance provisioning process is completed can update application configuration to refer to the endpoint of the new Encrypted database instance Once the database endpoint is changed at the application level, can remove the unencrypted instance.

From Command Line:

1. Run describe-db-instances command to list all RDS database names available in the selected AWS region, The command output should return the database instance identifier.

   ```
   aws rds describe-db-instances --region <region-name> --query 'DBInstances[*].DBInstanceIdentifier'
   ```
2. Run create-db-snapshot command to create a snapshot for the selected database instance, The command output will return the new snapshot with name DB Snapshot Name.

   ```
   aws rds create-db-snapshot --region <region-name> --db-snapshot-identifier <DB-Snapshot-Name> --db-instance-identifier <DB-Name>
   ```
3. Now run list-aliases command to list the KMS keys aliases available in a specified region, The command output should return each key alias currently available. For our RDS encryption activation process, locate the ID of the AWS default KMS key.

   ```
   aws kms list-aliases --region <region-name>
   ```
4. Run copy-db-snapshot command using the default KMS key ID for RDS instances returned earlier to create an encrypted copy of the database instance snapshot, The command output will return the encrypted instance snapshot Configuration.

   ```
   aws rds copy-db-snapshot --region <region-name> --source-db-snapshotidentifier <DB-Snapshot-Name> --target-db-snapshot-identifier <DB-SnapshotName-Encrypted> --copy-tags --kms-key-id <KMS-ID-For-RDS>
   ```
5. Run restore-db-instance-from-db-snapshot command to restore the encrypted snapshot created at the previous step to a new database instance, If successful, the command output should return the new encrypted database instance Configuration.

   ```
   aws rds restore-db-instance-from-db-snapshot --region <region-name> --dbinstance-identifier <DB-Name-Encrypted> --db-snapshot-identifier <DBSnapshot-Name-Encrypted>
   ```
6. Run describe-db-instances command to list all RDS database names, available in the selected AWS region, Output will return database instance identifier name Select encrypted database name that we just created DB-Name-Encrypted.

   ```
   aws rds describe-db-instances --region <region-name> --query 'DBInstances[*].DBInstanceIdentifier'
   ```
7. Run again describe-db-instances command using the RDS instance identifier returned earlier, to determine if the selected database instance is encrypted, The command output should return the encryption status True.

   ```
   aws rds describe-db-instances --region <region-name> --db-instance-identifier <DB-Name-Encrypted> --query 'DBInstances[*].StorageEncrypted'
   ```

Control: Ensure Auto Minor Version Upgrade feature is Enabled for RDS Instances

Sections: 2.3.2 4 55

Description: Ensure that RDS database instances have the Auto Minor Version Upgrade flag enabled in order to receive automatically minor engine upgrades during the specified maintenance window. So, RDS instances can get the new features, bug fixes, and security patches for their database engines.

Remediation Steps:
From Console:

1. Log in to the AWS management console and navigate to the RDS dashboard at https://console.aws.amazon.com/rds/.
2. In the left navigation panel, click on Databases.
3. Select the RDS instance that wants to update.
4. Click on the Modify button placed on the top right side.
5. On the Modify DB Instance: <instance identifier> page, In the Maintenance section, select Auto minor version upgrade click on the Yes radio button.
6. At the bottom of the page click on Continue, check to Apply Immediately to apply the changes immediately, or select Apply during the next scheduled maintenance window to avoid any downtime.
7. Review the changes and click on Modify DB Instance. The instance status should change from available to modifying and back to available. Once the feature is enabled, the Auto Minor Version Upgrade status should change to Yes.

From Command Line:

1. Run describe-db-instances command to list all RDS database instance names, available in the selected AWS region:

   ```
   aws rds describe-db-instances --region <regionName> --query 'DBInstances[*].DBInstanceIdentifier'
   ```
2. The command output should return each database instance identifier.
3. Run the modify-db-instance command to modify the selected RDS instance configuration this command will apply the changes immediately, Remove -- apply-immediately to apply changes during the next scheduled maintenance window and avoid any downtime:

   ```
   aws rds modify-db-instance --region <regionName> --db-instance-identifier <dbInstanceIdentifier> --auto-minor-version-upgrade –apply-immediately
   ```
4. The command output should reveal the new configuration metadata for the RDS instance and check AutoMinorVersionUpgrade parameter value.
5. Run describe-db-instances command to check if the Auto Minor Version Upgrade feature has been successfully enable:

   ```
   aws rds describe-db-instances --region <regionName> --db-instance-identifier <dbInstanceIdentifier> --query 'DBInstances[*].AutoMinorVersionUpgrade'
   ```
6. The command output should return the feature current status set to true, the feature is enabled and the minor engine upgrades will be applied to the selected RDS instance.

Control: Ensure that public access is not given to RDS Instance

Sections: 2.3.3 8 51

Description: Ensure and verify that RDS database instances provisioned in your AWS account do restrict unauthorized access in order to minimize security risks. To restrict access to any publicly accessible RDS database instance, you must disable the database Publicly Accessible flag and update the VPC security group associated with the instance.

Remediation Steps:
From Console:

1. Log in to the AWS management console and navigate to the RDS dashboard at https://console.aws.amazon.com/rds/.
2. Under the navigation panel, On RDS Dashboard, click Databases.
3. Select the RDS instance that you want to update.
4. Click Modify from the dashboard top menu.
5. On the Modify DB Instance panel, under the Connectivity section, click on Additional connectivity configuration and update the value for Publicly Accessible to Not publicly accessible to restrict public access. Follow the below steps to update subnet configurations:
   - Select the Connectivity and security tab, and click on the VPC attribute value inside the Networking section.
   - Select the Details tab from the VPC dashboard bottom panel and click on Route table configuration attribute value.
   - On the Route table details page, select the Routes tab from the dashboard bottom panel and click on Edit routes.
   - On the Edit routes page, update the Destination of Target which is set to igwxxxxx and click on Save routes.
6. On the Modify DB Instance panel Click on Continue and In the Scheduling of modifications section, perform one of the following actions based on your Requirements:
   - Select Apply during the next scheduled maintenance window to apply the changes automatically during the next scheduled maintenance window.
   - Select Apply immediately to apply the changes right away. With this option, any pending modifications will be asynchronously applied as soon as possible, regardless of the maintenance window setting for this RDS database instance. Note that any changes available in the pending modifications queue are also applied. If any of the pending modifications require downtime, choosing this option can cause unexpected downtime for the application.
7. Repeat steps 3 to 6 for each RDS instance available in the current region.
8. Change the AWS region from the navigation bar to repeat the process for other Regions.

From Command Line:

1. Run describe-db-instances command to list all RDS database names identifiers, available in the selected AWS region:
   ```
   aws rds describe-db-instances --region <region-name> --query 'DBInstances[*].DBInstanceIdentifier'
   ```
2. The command output should return each database instance identifier.
3. Run modify-db-instance command to modify the selected RDS instance configuration. Then use the following command to disable the Publicly Accessible flag for the selected RDS instances. This command use the applyimmediately flag. If you want to avoid any downtime --no-apply-immediately flag can be used:
   ```
   Aws rds modify-db-instance --region <region-name> --db-instance-identifier <db-name> --no-publicly-accessible –apply-immediately
   ```
4. The command output should reveal the PubliclyAccessible configuration under pending values and should get applied at the specified time.
5. Updating the Internet Gateway Destination via AWS CLI is not currently supported To update information about Internet Gateway use the AWS Console Procedure.
6. Repeat steps 1 to 5 for each RDS instance provisioned in the current region.
7. Change the AWS region by using the --region filter to repeat the process for other regions.

Control: Ensure that encryption is enabled for EFS file systems

Sections: 2.4.1 2 5

Description: EFS data should be encrypted at rest using AWS KMS (Key Management Service).

Remediation Steps:

It is important to note that EFS file system data at rest encryption must be turned on when creating the file system.

If an EFS file system has been created without data at rest encryption enabled then you must create another EFS file system with the correct configuration and transfer the data.

Steps to create an EFS file system with data encrypted at rest:

From Console:

1. Login to the AWS Management Console and Navigate to Elastic File System (EFS) dashboard.
2. Select File Systems from the left navigation panel.
3. Click Create File System button from the dashboard top menu to start the file system setup process.
4. On the Configure file system access configuration page, perform the following Actions.
   - Choose the right VPC from the VPC dropdown list.
   - Within Create mount targets section, select the checkboxes for all of the Availability Zones (AZs) within the selected VPC. These will be your mount Targets.
   - Click Next step to continue.
5. Perform the following on the Configure optional settings page.
   - Create tags to describe your new file system.
   - Choose performance mode based on your requirements.
   - Check Enable encryption checkbox and choose aws/elasticfilesystem from Select KMS master key dropdown list to enable encryption for the new file system using the default master key provided and managed by AWS KMS.
   - Click Next step to continue.
6. Review the file system configuration details on the review and create page and then click Create File System to create your new AWS EFS file system.
7. Copy the data from the old unencrypted EFS file system onto the newly create encrypted file system.
8. Remove the unencrypted file system as soon as your data migration to the newly create encrypted file system is completed.
9. Change the AWS region from the navigation bar and repeat the entire process for other aws regions.

From CLI:

1. Run describe-file-systems command to describe the configuration information available for the selected (unencrypted) file system (see Audit section to identify the right resource):

   ```
   aws efs describe-file-systems --region <region> --file-system-id <filesystem-id from audit section step 2 output>
   ```
2. The command output should return the requested configuration information.
3. To provision a new AWS EFS file system, you need to generate a universally unique identifier (UUID) in order to create the token required by the create-filesystem command. To create the required token, you can use a randomly generated UUID from "https://www.uuidgenerator.net".
4. Run create-file-system command using the unique token created at the previous Step.

   ```
   aws efs create-file-system --region <region> --creation-token <Token(randomly generated UUID from step 3)> --performance-mode generalPurpose –encrypted
   ```
5. The command output should return the new file system configuration metadata.
6. Run create-mount-target command using the newly created EFS file system ID returned at the previous step as identifier and the ID of the Availability Zone (AZ) that will represent the mount target:

   ```
   aws efs create-mount-target --region <region> --file-system-id <file-systemid> --subnet-id <subnet-id>
   ```
7. The command output should return the new mount target metadata.
8. Now you can mount your file system from an EC2 instance.
9. Copy the data from the old unencrypted EFS file system onto the newly create encrypted file system.
10. Remove the unencrypted file system as soon as your data migration to the newly create encrypted file system is completed.

    ```
    aws efs delete-file-system --region <region> --file-system-id <unencryptedfile-system-id>
    ```
11. Change the AWS region by updating the --region and repeat the entire process for other aws regions.

Control: Ensure CloudTrail log file validation is enabled

Sections: 3.2 20 19

Description: CloudTrail log file validation creates a digitally signed digest file containing a hash of each log that CloudTrail writes to S3. These digest files can be used to determine whether a log file was changed, deleted, or unchanged after CloudTrail delivered the log. It is recommended that file validation be enabled on all CloudTrails.

Remediation Steps:
Perform the following to enable log file validation on a given trail:

From Console:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/cloudtrail
2. Click on Trails on the left navigation pane
3. Click on target trail
4. Within the General details section click edit
5. Under the Advanced settings section
6. Check the enable box under Log file validation
7. Click Save changes

From Command Line:

```
aws cloudtrail update-trail --name <trail_name> --enable-log-file-validation
```

Note that periodic validation of logs using these digests can be performed by running the following command:

```
aws cloudtrail validate-logs --trail-arn <trail_arn> --start-time <start_time> --end-time <end_time>
```

Control: Ensure CloudTrail trails are integrated with CloudWatch Logs

Sections: 3.4 39 0

Description: AWS CloudTrail is a web service that records AWS API calls made in a given AWS account. The recorded information includes the identity of the API caller, the time of the API call, the source IP address of the API caller, the request parameters, and the response elements returned by the AWS service. CloudTrail uses Amazon S3 for log file storage and delivery, so log files are stored durably. In addition to capturing CloudTrail logs within a specified S3 bucket for long term analysis, real time analysis can be performed by configuring CloudTrail to send logs to CloudWatch Logs. For a trail that is enabled in all regions in an account, CloudTrail sends log files from all those regions to a CloudWatch Logs log group. It is recommended that CloudTrail logs be sent to CloudWatch Logs. Note: The intent of this recommendation is to ensure AWS account activity is being captured, monitored, and appropriately alarmed on. CloudWatch Logs is a native way to accomplish this using AWS services but does not preclude the use of an alternate solution.

Remediation Steps:
Perform the following to establish the prescribed state:

From Console:

1. Login to the CloudTrail console at https://console.aws.amazon.com/cloudtrail/
2. Select the Trail the needs to be updated.
3. Scroll down to CloudWatch Logs
4. Click Edit
5. Under CloudWatch Logs click the box Enabled
6. Under Log Group pick new or select an existing log group
7. Edit the Log group name to match the CloudTrail or pick the existing CloudWatch Group.
8. Under IAM Role pick new or select an existing.
9. Edit the Role name to match the CloudTrail or pick the existing IAM Role. 10.Click **Save changes**.

From Command Line:

```
aws cloudtrail update-trail --name <trail_name> --cloudwatch-logs-log-grouparn <cloudtrail_log_group_arn> --cloudwatch-logs-role-arn <cloudtrail_cloudwatchLogs_role_arn>
```

Control: Ensure AWS Config is enabled in all regions

Sections: 3.5 323 0

Description: AWS Config is a web service that performs configuration management of supported AWS resources within your account and delivers log files to you. The recorded information includes the configuration item (AWS resource), relationships between configuration items (AWS resources), any configuration changes between resources. It is recommended AWS Config be enabled in all regions.

Remediation Steps:
To implement AWS Config configuration:

From Console:

1. Select the region you want to focus on in the top right of the console
2. Click Services
3. Click Config
4. If a Config recorder is enabled in this region, you should navigate to the Settings page from the navigation menu on the left hand side. If a Config recorder is not yet enabled in this region then you should select "Get Started".
5. Select "Record all resources supported in this region"
6. Choose to include global resources (IAM resources)
7. Specify an S3 bucket in the same account or in another managed AWS account
8. Create an SNS Topic from the same AWS account or another managed AWS Account

From Command Line:

1. Ensure there is an appropriate S3 bucket, SNS topic, and IAM role per the AWS Config Service prerequisites.
2. Run this command to create a new configuration recorder:

```
aws configservice put-configuration-recorder --configuration-recorder
name=default,roleARN=arn:aws:iam::012345678912:role/myConfigRole --recordinggroup
allSupported=true,includeGlobalResourceTypes=true
```

3. Create a delivery channel configuration file locally which specifies the channel attributes, populated from the prerequisites set up previously:

```
{ "name": "default", "s3BucketName": "my-config-bucket", "snsTopicARN": "arn:aws:sns:us-east-1:012345678912:my-config-notice",
"configSnapshotDeliveryProperties": { "deliveryFrequency": "Twelve_Hours" } }
```

4. Run this command to create a new delivery channel, referencing the json configuration file made in the previous step:

```
aws configservice put-delivery-channel --delivery-channel file://deliveryChannel.json
```

5. Start the configuration recorder by running the following command:

```
aws configservice start-configuration-recorder --configuration-recorder-name default
```

Control: Ensure S3 bucket access logging is enabled on the CloudTrail S3 bucket

Sections: 3.6 39 0

Description: S3 Bucket Access Logging generates a log that contains access records for each request made to your S3 bucket. An access log record contains details about the request, such as the request type, the resources specified in the request worked, and the time and date the request was processed. It is recommended that bucket access logging be enabled on the CloudTrail S3 bucket.

Remediation Steps:
Perform the following to enable S3 bucket logging:

From Console:

1. Sign in to the AWS Management Console and open the S3 console at https://console.aws.amazon.com/s3.
2. Under All Buckets click on the target S3 bucket
3. Click on Properties in the top right of the console
4. Under Bucket: <s3_bucket_for_cloudtrail> click on Logging
5. Configure bucket logging
   - Click on the Enabled checkbox
   - Select Target Bucket from list
   - Enter a Target Prefix
6. Click Save.

From Command Line:

1. Get the name of the S3 bucket that CloudTrail is logging to:

   ```
   aws cloudtrail describe-trails --region <region-name> --query trailList[\*].S3BucketName
   ```

2. Copy and add target bucket name at <Logging_BucketName>, Prefix for logfile at <LogFilePrefix> and optionally add an email address in the following template and save it as <FileName.Json>:

   ```
   { "LoggingEnabled": { "TargetBucket": "<Logging_BucketName>", "TargetPrefix": "<LogFilePrefix>", "TargetGrants": [ { "Grantee": { "Type": "AmazonCustomerByEmail", "EmailAddress": "<EmailID>" }, "Permission": "FULL_CONTROL" } ] } }
   ```

3. Run the put-bucket-logging command with bucket name and <FileName.Json> as input, for more information refer at put-bucket-logging:

   ```
   Aws s3api put-bucket-logging --bucket <BucketName> --bucket-logging-status file://<FileName.Json>
   ```

Control: Ensure CloudTrail logs are encrypted at rest using KMS CMKs

Sections: 3.7 38 1

Description: AWS CloudTrail is a web service that records AWS API calls for an account and makes those logs available to users and resources in accordance with IAM policies. AWS Key Management Service (KMS) is a managed service that helps create and control the encryption keys used to encrypt account data, and uses Hardware Security Modules (HSMs) to protect the security of encryption keys. CloudTrail logs can be configured to leverage server side encryption (SSE) and KMS customer created master keys (CMK) to further protect CloudTrail logs. It is recommended that CloudTrail be configured to use SSE-KMS.

Remediation Steps:
Perform the following to configure CloudTrail to use SSE-KMS:

From Console:

1. Sign in to the AWS Management Console and open the CloudTrail console at https://console.aws.amazon.com/cloudtrail
2. In the left navigation pane, choose Trails .
3. Click on a Trail
4. Under the S3 section click on the edit button (pencil icon)
5. Click Advanced
6. Select an existing CMK from the KMS key Id drop-down menu
   - Note: Ensure the CMK is located in the same region as the S3 bucket
   - Note: You will need to apply a KMS Key policy on the selected CMK in order for CloudTrail as a service to encrypt and decrypt log files using the CMK provided. Steps are provided here for editing the selected CMK Key policy
7. Click Save
8. You will see a notification message stating that you need to have decrypt permissions on the specified KMS key to decrypt log files.
9. Click Yes

From Command Line:

```
aws cloudtrail update-trail --name <trail_name> --kms-id <cloudtrail_kms_key> aws kms put-key-policy --key-id <cloudtrail_kms_key> --policy <cloudtrail_kms_key_policy>
```

Control: Ensure rotation for customer created symmetric CMKs is enabled

Sections: 3.8 25 10

Description: AWS Key Management Service (KMS) allows customers to rotate the backing key which is key material stored within the KMS which is tied to the key ID of the Customer Created customer master key (CMK). It is the backing key that is used to perform cryptographic operations such as encryption and decryption. Automated key rotation currently retains all prior backing keys so that decryption of encrypted data can take place transparently. It is recommended that CMK key rotation be enabled for symmetric keys. Key rotation can not be enabled for any asymmetric CMK.

Remediation Steps:
From Console:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam.
2. In the left navigation pane, choose Customer managed keys .
3. Select a customer managed CMK where Key spec = SYMMETRIC_DEFAULT
4. Underneath the "General configuration" panel open the tab "Key rotation"
5. Check the "Automatically rotate this KMS key every year." checkbox

From Command Line:

1. Run the following command to enable key rotation:

```
aws kms enable-key-rotation --key-id <kms_key_id>
```

Control: Ensure VPC flow logging is enabled in all VPCs

Sections: 3.9 422 5

Description: VPC Flow Logs is a feature that enables you to capture information about the IP traffic going to and from network interfaces in your VPC. After you've created a flow log, you can view and retrieve its data in Amazon CloudWatch Logs. It is recommended that VPC Flow Logs be enabled for packet "Rejects" for VPCs.

Remediation Steps:
Perform the following to determine if VPC Flow logs is enabled:

From Console:

1. Sign into the management console
2. Select Services then VPC
3. In the left navigation pane, select Your VPCs
4. Select a VPC
5. In the right pane, select the Flow Logs tab.
6. If no Flow Log exists, click Create Flow Log
7. For Filter, select Reject
8. Enter in a Role and Destination Log Group
9. Click Create Log Flow 10.Click on CloudWatch Logs Group

Note: Setting the filter to "Reject" will dramatically reduce the logging data accumulation for this recommendation and provide sufficient information for the purposes of breach detection, research and remediation. However, during periods of least privilege security group engineering, setting this the filter to "All" can be very helpful in discovering existing traffic flows required for proper operation of an already running environment.

From Command Line:

1. Create a policy document and name it as role_policy_document.json and paste the following content:

   ```
   { "Version": "2012-10-17", "Statement": [ { "Sid": "test", "Effect": "Allow", "Principal": { "Service": "ec2.amazonaws.com" }, "Action": "sts:AssumeRole" } ] }
   ```

2. Create another policy document and name it as iam_policy.json and paste the following content:

   ```
   { "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action":[ "logs:CreateLogGroup", "logs:CreateLogStream", "logs:DescribeLogGroups", "logs:DescribeLogStreams", "logs:PutLogEvents", "logs:GetLogEvents", "logs:FilterLogEvents" ], "Resource": "*" } ] }
   ```

3. Run the below command to create an IAM role:

   ```
   aws iam create-role --role-name <aws_support_iam_role> --assume-role-policydocument file://<file-path>role_policy_document.json
   ```

4. Run the below command to create an IAM policy:

   ```
   aws iam create-policy --policy-name <ami-policy-name> --policy-document file://<file-path>iam-policy.json
   ```

5. Run attach-group-policy command using the IAM policy ARN returned at the previous step to attach the policy to the IAM role (if the command succeeds, no output is returned):

   ```
   aws iam attach-group-policy --policy-arn arn:aws:iam::<aws-accountid>:policy/<iam-policy-name> --group-name <group-name>
   ```

6. Run describe-vpcs to get the VpcId available in the selected region:

   ```
   aws ec2 describe-vpcs --region <region>
   ```

7. The command output should return the VPC Id available in the selected region.
8. Run create-flow-logs to create a flow log for the vpc:

   ```
   aws ec2 create-flow-logs --resource-type VPC --resource-ids <vpc-id> --traffic-type REJECT --log-group-name <log-group-name> --deliver-logspermission-arn <iam-role-arn>
   ```

9. Repeat step 8 for other vpcs available in the selected region. 10.Change the region by updating --region and repeat remediation procedure for other vpcs.

Control: Ensure that Object-level logging for write events is enabled for S3 bucket

Sections: 3.10 706 0

Description: S3 object-level API operations such as GetObject, DeleteObject, and PutObject are called data events. By default, CloudTrail trails don't log data events and so it is recommended to enable Object-level logging for S3 buckets.

Remediation Steps:
From Console:

1. Login to the AWS Management Console and navigate to S3 dashboard at https://console.aws.amazon.com/s3/
2. In the left navigation panel, click buckets and then click on the S3 Bucket Name that you want to examine.
3. Click Properties tab to see in detail bucket configuration.
4. Click on the Object-level logging setting, enter the CloudTrail name for the recording activity. You can choose an existing Cloudtrail or create a new one by navigating to the Cloudtrail console link https://console.aws.amazon.com/cloudtrail/
5. Once the Cloudtrail is selected, check the Write event checkbox, so that objectlevel logging for Write events is enabled.
6. Repeat steps 2 to 5 to enable object-level logging of write events for other S3 Buckets.

From Command Line:

1. To enable object-level data events logging for S3 buckets within your AWS account, run put-event-selectors command using the name of the trail that you want to reconfigure as identifier:

   ```
   aws cloudtrail put-event-selectors --region <region-name> --trail-name <trail-name> --event-selectors '[{ "ReadWriteType":"WriteOnly","IncludeManagementEvents":true,"DataResources": [{ "Type":"AWS::S3::Object", "Values": ["arn:aws:s3:::<s3-bucket-name>/"] }] }]'
   ```
2. The command output will be object-level event trail configuration.
3. If you want to enable it for all buckets at once then change Values parameter to ["arn:aws:s3"] in command given above.
4. Repeat step 1 for each s3 bucket to update object-level logging of write events.
5. Change the AWS region by updating the --region command parameter and perform the process for other regions.

Control: Ensure that Object-level logging for read events is enabled for S3 bucket

Sections: 3.11 706 0

Description: S3 object-level API operations such as GetObject, DeleteObject, and PutObject are called data events. By default, CloudTrail trails don't log data events and so it is recommended to enable Object-level logging for S3 buckets.

Remediation Steps:
From Console:

1. Login to the AWS Management Console and navigate to S3 dashboard at https://console.aws.amazon.com/s3/
2. In the left navigation panel, click buckets and then click on the S3 Bucket Name that you want to examine.
3. Click Properties tab to see in detail bucket configuration.
4. Click on the Object-level logging setting, enter the CloudTrail name for the recording activity. You can choose an existing Cloudtrail or create a new one by navigating to the Cloudtrail console link https://console.aws.amazon.com/cloudtrail/
5. Once the Cloudtrail is selected, check the Read event checkbox, so that objectlevel logging for Read events is enabled.
6. Repeat steps 2 to 5 to enable object-level logging of read events for other S3 Buckets.

From Command Line:

1. To enable object-level data events logging for S3 buckets within your AWS account, run put-event-selectors command using the name of the trail that you want to reconfigure as identifier:

   ```
   aws cloudtrail put-event-selectors --region <region-name> --trail-name <trail-name> --event-selectors '[{ "ReadWriteType": "ReadOnly","IncludeManagementEvents":true,"DataResources": [{ "Type":"AWS::S3::Object", "Values": ["arn:aws:s3:::<s3-bucket-name>/"] }] }]'
   ```
2. The command output will be object-level event trail configuration.
3. If you want to enable it for all buckets at once then change Values parameter to ["arn:aws:s3"] in command given above.
4. Repeat step 1 for each s3 bucket to update object-level logging of read events.
5. Change the AWS region by updating the --region command parameter and perform the process for other regions.

Control: Ensure unauthorized API calls are monitored

Sections: 4.1 19 0

Description: Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs, or an external Security information and event management (SIEM) environment, and establishing corresponding metric filters and alarms. It is recommended that a metric filter and alarm be established for unauthorized API calls.

Remediation Steps:
If you are using CloudTrails and CloudWatch, perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for unauthorized API calls and the <cloudtrail_log_group_name> taken from audit Step 1.

   ```
   aws logs put-metric-filter --log-group-name "cloudtrail_log_group_name" --filter-name "<unauthorized_api_calls_metric>" --metric-transformationsmetricName=unauthorized_api_calls_metric,metricNamespace=CISBenchmark,metricValue=1 --filter-pattern "{ ($.errorCode ="*UnauthorizedOperation") ||($.errorCode ="AccessDenied*") &&($.sourceIPAddress!="delivery.logs.amazonaws.com") && ($.eventName!="HeadBucket") }"
   ```

   Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.
2. Create an SNS topic that the alarm will notify

   ```
   aws sns create-topic --name <sns_topic_name>
   ```

   Note: you can execute this command once and then re-use the same topic for all monitoring alarms.

   Note: Capture the TopicArn displayed when creating the SNS Topic in Step 2.
3. Create an SNS subscription to the topic created in step 2

   ```
   aws sns subscribe --topic-arn <sns_topic_arn from step 2> --protocol <protocol_for_sns> --notification-endpoint <sns_subscription_endpoints>
   ```

   Note: you can execute this command once and then re-use the SNS subscription for all monitoring alarms.
4. Create an alarm that is associated with the CloudWatch Logs Metric Filter created in step 1 and an SNS topic created in step 2

   ```
   aws cloudwatch put-metric-alarm --alarm-name "unauthorized_api_calls_alarm" --metric-name "unauthorized_api_calls_metric" --statistic Sum --period 300 --threshold 1 --comparison-operator GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace "CISBenchmark" --alarm-actions <sns_topic_arn>
   ```

Control: Ensure management console sign-in without MFA is monitored

Sections: 4.2 19 0

Description: Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs, or an external Security information and event management (SIEM) environment, and establishing corresponding metric filters and alarms. It is recommended that a metric filter and alarm be established for console logins that are not protected by multi-factor authentication (MFA)

Remediation Steps:
If you are using CloudTrails and CloudWatch, perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for AWS Management Console sign-in without MFA and the <cloudtrail_log_group_name> taken from audit step 1.

   Use Command:

   ```
   aws logs put-metric-filter --log-group-name <cloudtrail\_log\_group\_name> --filter-name `<no\_mfa\_console\_signin\_metric>` --metric-transformationsmetricName= `<no\_mfa\_console\_signin\_metric>`,metricNamespace='CISBenchmark',metricValue=1 --filter-pattern '{($.eventName = "ConsoleLogin") && ($.additionalEventData.MFAUsed != "Yes") }'
   ```

   Or (To reduce false positives incase Single Sign-On (SSO) is used in organization):

   ```
   aws logs put-metric-filter --log-group-name <cloudtrail\_log\_group\_name> --filter-name `<no\_mfa\_console\_signin\_metric>` --metric-transformations
   ```

metricName= `<no\_mfa\_console\_signin\_metric>`,metricNamespace='CISBenchmark',metricValue=1 --filter-pattern '{($.eventName = "ConsoleLogin") && ($.additionalEventData.MFAUsed != "Yes") &&($.userIdentity.type = "IAMUser") && ($.responseElements.ConsoleLogin ="Success") }' ```

```
Note: You can choose your own metricName and metricNamespace strings. Using the
```

same metricNamespace for all Foundations Benchmark metrics will group them together. 2. Create an SNS topic that the alarm will notify

```
aws sns create-topic --name <sns_topic_name>
```

```
Note: you can execute this command once and then re-use the same topic for all
```

monitoring alarms. 3. Create an SNS subscription to the topic created in step 2

```
aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --notification-endpoint <sns_subscription_endpoints>
```

```
Note: you can execute this command once and then re-use the SNS subscription for all
```

monitoring alarms. 4. Create an alarm that is associated with the CloudWatch Logs Metric Filter created in step 1 and an SNS topic created in step 2

```
aws cloudwatch put-metric-alarm --alarm-name `<no_mfa_console_signin_alarm>` --metric-name `<no_mfa_console_signin_metric>` --statis
```

Control: Ensure usage of 'root' account is monitored

Sections: 4.3 19 0

Description: Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs, or an external Security information and event management (SIEM) environment, and establishing corresponding metric filters and alarms. It is recommended that a metric filter and alarm be established for 'root' login attempts to detect the unauthorized use, or attempts to use the root account.

Remediation Steps:
If you are using CloudTrails and CloudWatch, perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for 'Root' account usage and the <cloudtrail_log_group_name> taken from audit step 1.

   ```
   aws logs put-metric-filter --log-group-name `<cloudtrail_log_group_name>` --filter-name `<root_usage_metric>` --metric-transformations metricName=`<root_usage_metric>` ,metricNamespace='CISBenchmark',metricValue=1 --filterpattern '{ $.userIdentity.type = "Root" && $.userIdentity.invokedBy NOTEXISTS && $.eventType !="AwsServiceEvent" }'
   ```

   Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.
2. Create an SNS topic that the alarm will notify

   ```
   aws sns create-topic --name <sns_topic_name>
   ```

   Note: you can execute this command once and then re-use the same topic for all monitoring alarms.
3. Create an SNS subscription to the topic created in step 2

   ```
   aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --notification-endpoint <sns_subscription_endpoints>
   ```

   Note: you can execute this command once and then re-use the SNS subscription for all monitoring alarms.
4. Create an alarm that is associated with the CloudWatch Logs Metric Filter created in step 1 and an SNS topic created in step 2

   ```
   aws cloudwatch put-metric-alarm --alarm-name `<root_usage_alarm>` --metricname `<root_usage_metric>` --statistic Sum --period 300 --threshold 1 --comparison-operator GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace 'CISBenchmark' --alarm-actions <sns_topic\_arn>
   ```

Control: Ensure IAM policy changes are monitored

Sections: 4.4 19 0

Description: Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs, or an external Security information and event management (SIEM) environment, and establishing corresponding metric filters and alarms. It is recommended that a metric filter and alarm be established changes made to Identity and Access Management (IAM) policies.

Remediation Steps:
If you are using CloudTrails and CloudWatch, perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for IAM policy changes and the <cloudtrail_log_group_name> taken from audit step 1.

aws logs put-metric-filter --log-group-name `<cloudtrail_log_group_name>` --filter-name `<iam_changes_metric>` --metric-transformations metricName=`<iam_changes_metric>` ,metricNamespace='CISBenchmark',metricValue=1 --filter-pattern'{($.eventName=DeleteGroupPolicy)||($.eventName=DeleteRolePolicy)||($.eventName=DeleteUserPolicy)|| ($.eventName=PutGroupPolicy)||($.eventName=PutRolePolicy)||($.eventName=PutUserPolicy)||($.eventName=CreatePolicy)|| ($.eventName=DeletePolicy)||($.eventName=CreatePolicyVersion)||($.eventName=DeletePolicyVersion)|| ($.eventName=AttachRolePolicy)||($.eventName=DetachRolePolicy)||($.eventName=AttachUserPolicy)|| ($.eventName=DetachUserPolicy)||($.eventName=AttachGroupPolicy)||($.eventName=DetachGroupPolicy)}' ```

Note: You can choose your own metricName and metricNamespace strings. Using the

same metricNamespace for all Foundations Benchmark metrics will group them together. 2. Create an SNS topic that the alarm will notify

```
aws sns create-topic --name <sns_topic_name>
```

Note: you can execute this command once and then re-use the same topic for all

monitoring alarms. 3. Create an SNS subscription to the topic created in step 2

```
aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --notification-endpoint <sns_subscription_endpoints>
```

Note: you can execute this command once and then re-use the SNS subscription for all

monitoring alarms. 4. Create an alarm that is associated with the CloudWatch Logs Metric Filter created in step 1 and an SNS topic created in step 2

```
aws cloudwatch put-metric-alarm --alarm-name `<iam_changes_alarm>` --metric-name `<iam_changes_metric>` --statistic Sum --period 300
```

Control: Ensure CloudTrail configuration changes are monitored

Sections: 4.5 19 0

Description: Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs, or an external Security information and event management (SIEM) environment, where metric filters and alarms can be established. It is recommended that a metric filter and alarm be utilized for detecting changes to CloudTrail's configurations.

Remediation Steps:
If you are using CloudTrails and CloudWatch, perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for cloudtrail configuration changes and the <cloudtrail_log_group_name> taken from audit Step 1.

```
aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name `<cloudtrail_cfg_changes_metric>` --
metric-transformationsmetricName= `<cloudtrail_cfg_changes_metric>`,metricNamespace='CISBenchmark',metricValue=1 --filter-
pattern '{($.eventName = CreateTrail) || ($.eventName = UpdateTrail) || ($.eventName =
```

DeleteTrail) || ($.eventName = StartLogging) || ($.eventName = StopLogging)}' ```
Note: You can choose your own metricName and metricNamespace strings. Using the
same metricNamespace for all Foundations Benchmark metrics will group them together. 2. Create an SNS topic that the alarm will notify
```
aws sns create-topic --name <sns_topic_name>
```

Note: you can execute this command once and then re-use the same topic for all
monitoring alarms. 3. Create an SNS subscription to the topic created in step 2
```
aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --notification-endpoint <sns_subscription_endpoints>
```

Note: you can execute this command once and then re-use the SNS subscription for all
monitoring alarms. 4. Create an alarm that is associated with the CloudWatch Logs Metric Filter created in step 1 and an SNS topic created in step 2
```
aws cloudwatch put-metric-alarm --alarm-name<cloudtrail_cfg_changes_alarm> --metric-name <cloudtrail_cfg_changes_metric> --statistic
Sum --period 300 --threshold 1 --comparison-operator GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace
'CISBenchmark' --alarm-actions <sns_topic_arn> ```
```

Control: Ensure AWS Management Console authentication failures are monitored

Sections: 4.6 19 0

Description: Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs, or an external Security information and event management (SIEM) environment, and establishing corresponding metric filters and alarms. It is recommended that a metric filter and alarm be established for failed console authentication attempts.

Remediation Steps:
If you are using CloudTrails and CloudWatch, perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for AWS management Console Login Failures and the <cloudtrail_log_group_name> taken from audit step 1.

   ```
   aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name `<console_signin_failure_metric>` --metric-transformationsmetricName= `<console_signin_failure_metric>`,metricNamespace='CISBenchmark',metricValue=1 --filter-pattern '{($.eventName = ConsoleLogin) && ($.errorMessage = "Failedauthentication") }'
   ```

   Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.
2. Create an SNS topic that the alarm will notify

   ```
   aws sns create-topic --name <sns_topic_name>
   ```

   Note: you can execute this command once and then re-use the same topic for all monitoring alarms.
3. Create an SNS subscription to the topic created in step 2

   ```
   aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --notification-endpoint <sns_subscription_endpoints>
   ```

   Note: you can execute this command once and then re-use the SNS subscription for all monitoring alarms.
4. Create an alarm that is associated with the CloudWatch Logs Metric Filter created in step 1 and an SNS topic created in step 2

   ```
   aws cloudwatch put-metric-alarm --alarm-name `<console_signin_failure_alarm>` --metric-name `<console_signin_failure_metric>` --statistic Sum --period 300 --threshold 1 --comparison-operator GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace 'CISBenchmark' --alarm-actions <sns_topic_arn>
   ```

Control: Ensure disabling or scheduled deletion of customer created CMKs is monitored

Sections: 4.7 19 0

Description: Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs, or an external Security information and event management (SIEM) environment, and establishing corresponding metric filters and alarms. It is recommended that a metric filter and alarm be established for customer created CMKs which have changed state to disabled or scheduled deletion.

Remediation Steps:
If you are using CloudTrails and CloudWatch, perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for disabled or scheduled for deletion CMK's and the <cloudtrail_log_group_name> taken from audit step 1.

   ```
   aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name `<disable_or_delete_cmk_changes_metric>` --metrictransformationsmetricName= `<disable_or_delete_cmk_changes_metric>`,metricNamespace='CISBenchmark',metricValue=1 --filter-pattern '{($.eventSource = kms.amazonaws.com) && (($.eventName=DisableKey)||($.eventName=ScheduleKeyDeletion)) }'
   ```

   Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.
2. Create an SNS topic that the alarm will notify

   ```
   aws sns create-topic --name <sns_topic_name>
   ```

   Note: you can execute this command once and then re-use the same topic for all monitoring alarms.
3. Create an SNS subscription to the topic created in step 2

   ```
   aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --notification-endpoint <sns_subscription_endpoints>
   ```

   Note: you can execute this command once and then re-use the SNS subscription for all monitoring alarms.
4. Create an alarm that is associated with the CloudWatch Logs Metric Filter created in step 1 and an SNS topic created in step 2

   ```
   aws cloudwatch put-metric-alarm --alarm-name `<disable_or_delete_cmk_changes_alarm>` --metric-name `<disable_or_delete_cmk_changes_metric>` --statistic Sum --period 300 --threshold 1 --comparison-operator GreaterThanOrEqualToThreshold --evaluationperiods 1 --namespace 'CISBenchmark' --alarm-actions <sns_topic_arn>
   ```

Control: Ensure S3 bucket policy changes are monitored

Sections: 4.8 19 0

Description: Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs, or an external Security information and event management (SIEM) environment, and establishing corresponding metric filters and alarms. It is recommended that a metric filter and alarm be established for changes to S3 bucket policies.

Remediation Steps:
If you are using CloudTrails and CloudWatch, perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for S3 bucket policy changes and the <cloudtrail_log_group_name> taken from audit step 1.

```
aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name `<s3_bucket_policy_changes_metric>` --metric-transformationsmetricName= `<s3_bucket_policy_changes_metric>`,metricNamespace='CISBenchmark',metricValue=1 --filter-pattern '{($.eventSource = s3.amazonaws.com) && (($.eventName = PutBucketAcl) ||
```

($.eventName = PutBucketPolicy) || ($.eventName = PutBucketCors) || ($.eventName = PutBucketLifecycle) || ($.eventName = PutBucketReplication) || ($.eventName = DeleteBucketPolicy) || ($.eventName = DeleteBucketCors) || ($.eventName = DeleteBucketLifecycle) || ($.eventName = DeleteBucketReplication)) }' ```

```
Note: You can choose your own metricName and metricNamespace strings. Using the
```

same metricNamespace for all Foundations Benchmark metrics will group them together. 2. Create an SNS topic that the alarm will notify

```
aws sns create-topic --name <sns_topic_name>
```

```
Note: you can execute this command once and then re-use the same topic for all
```

monitoring alarms. 3. Create an SNS subscription to the topic created in step 2

```
aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --notification-endpoint <sns_subscription_endpoints>
```

```
Note: you can execute this command once and then re-use the SNS subscription for all
```

monitoring alarms. 4. Create an alarm that is associated with the CloudWatch Logs Metric Filter created in step 1 and an SNS topic created in step 2

```
aws cloudwatch put-metric-alarm --alarm-name `<s3_bucket_policy_changes_alarm>` --metric-name `<s3_bucket_policy_changes_metric>` --
```

Control: Ensure AWS Config configuration changes are monitored

Sections: 4.9 19 0

Description: Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs, or an external Security information and event management (SIEM) environment, and establishing corresponding metric filters and alarms. It is recommended that a metric filter and alarm be established for detecting changes to AWS Config's configurations.

Remediation Steps:
If you are using CloudTrails and CloudWatch, perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for AWS Configuration changes and the <cloudtrail_log_group_name> taken from audit Step 1.

   ```
   aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name `<aws_config_changes_metric>` --metric-transformationsmetricName= `<aws_config_changes_metric>`,metricNamespace='CISBenchmark',metricValue=1 --filter-pattern '{($.eventSource = config.amazonaws.com) && (($.eventName=StopConfigurationRecorder)||($.eventName=DeleteDeliveryChannel) || ($.eventName=PutDeliveryChannel)||($.eventName=PutConfigurationRecorder))}'
   ```

   Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.
2. Create an SNS topic that the alarm will notify

   ```
   aws sns create-topic --name <sns_topic_name>
   ```

   Note: you can execute this command once and then re-use the same topic for all monitoring alarms.
3. Create an SNS subscription to topic created in step 2

   ```
   aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --notification-endpoint <sns_subscription_endpoints>
   ```

   Note: you can execute this command once and then re-use the SNS subscription for all monitoring alarms.
4. Create an alarm that is associated with the CloudWatch Logs Metric Filter created in step 1 and an SNS topic created in step 2

aws cloudwatch put-metric-alarm --alarm-name <aws_config_changes_alarm> --metric-name <aws_config_changes_metric> --statistic Sum --period 300 --threshold 1 --comparison-operator GreaterThanOrEqualToThreshold --evaluationperiods 1 --namespace 'CISBenchmark' --alarm-actions <sns_topic_arn> ```

Control: Ensure security group changes are monitored

Sections: 4.10 19 0

Description: Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs, or an external Security information and event management (SIEM) environment, and establishing corresponding metric filters and alarms. Security Groups are a stateful packet filter that controls ingress and egress traffic within a VPC. It is recommended that a metric filter and alarm be established for detecting changes to Security Groups.

Remediation Steps:
If you are using CloudTrails and CloudWatch, perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for security groups changes and the <cloudtrail_log_group_name> taken from audit step 1.

```
aws logs put-metric-filter --log-group-name "<cloudtrail_log_group_name>" --filter-name "<security_group_changes_metric>" --
metric-transformationsmetricName= "<security_group_changes_metric>",metricNamespace="CISBenchmark",metricValue=1 --filter-
pattern "{($.eventName = AuthorizeSecurityGroupIngress) || ($.eventName =
```

AuthorizeSecurityGroupEgress) || ($.eventName = RevokeSecurityGroupIngress)|| ($.eventName = RevokeSecurityGroupEgress) || ($.eventName =CreateSecurityGroup) || ($.eventName = DeleteSecurityGroup) }" ```

```
Note: You can choose your own metricName and metricNamespace strings. Using the
```

same metricNamespace for all Foundations Benchmark metrics will group them together. 2. Create an SNS topic that the alarm will notify

```
aws sns create-topic --name "<sns_topic_name>"
```

```
Note: you can execute this command once and then re-use the same topic for all
```

monitoring alarms. 3. Create an SNS subscription to the topic created in step 2

```
aws sns subscribe --topic-arn "<sns_topic_arn>" --protocol <protocol_for_sns> --notification-endpoint "<sns_subscription_endpoints>"
```

```
Note: you can execute this command once and then re-use the SNS subscription for all
```

monitoring alarms. 4. Create an alarm that is associated with the CloudWatch Logs Metric Filter created in step 1 and an SNS topic created in step 2

```
aws cloudwatch put-metric-alarm --alarm-name "<security_group_changes_alarm>" --metric-name "<security_group_changes_metric>" --stat
```

Control: Ensure Network Access Control Lists (NACL) changes are monitored

Sections: 4.11 19 0

Description: Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs, or an external Security information and event management (SIEM) environment, and establishing corresponding metric filters and alarms. NACLs are used as a stateless packet filter to control ingress and egress traffic for subnets within a VPC. It is recommended that a metric filter and alarm be established for changes made to NACLs.

Remediation Steps:
If you are using CloudTrails and CloudWatch, perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for NACL changes and the <cloudtrail_log_group_name> taken from audit step 1.

   ```
   aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name `<nacl_changes_metric>` --metric-transformations metricName=`<nacl_changes_metric>` ,metricNamespace='CISBenchmark',metricValue=1 --filter-pattern '{ ($.eventName = CreateNetworkAcl) || ($.eventName =CreateNetworkAclEntry) || ($.eventName = DeleteNetworkAcl) || ($.eventName =DeleteNetworkAclEntry) || ($.eventName = ReplaceNetworkAclEntry) ||($.eventName = ReplaceNetworkAclAssociation) }'
   ```

   Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.
2. Create an SNS topic that the alarm will notify

   ```
   aws sns create-topic --name <sns_topic_name>
   ```

   Note: you can execute this command once and then re-use the same topic for all monitoring alarms.
3. Create an SNS subscription to the topic created in step 2

   ```
   aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --notification-endpoint <sns_subscription_endpoints>
   ```

   Note: you can execute this command once and then re-use the SNS subscription for all monitoring alarms.
4. Create an alarm that is associated with the CloudWatch Logs Metric Filter created in step 1 and an SNS topic created in step 2

   ```
   aws cloudwatch put-metric-alarm --alarm-name `<nacl_changes_alarm>` --metric-name `<nacl_changes_metric>` --statistic Sum --period 300 --threshold 1 --comparison-operator GreaterThanOrEqualToThreshold --evaluationperiods 1 --namespace 'CISBenchmark' --alarm-actions <sns_topic_arn>
   ```

Control: Ensure changes to network gateways are monitored

Sections: 4.12 19 0

Description: Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs, or an external Security information and event management (SIEM) environment, and establishing corresponding metric filters and alarms. Network gateways are required to send/receive traffic to a destination outside of a VPC. It is recommended that a metric filter and alarm be established for changes to network

Remediation Steps:
If you are using CloudTrails and CloudWatch, perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for network gateways changes and the <cloudtrail_log_group_name> taken from audit step 1.

   `aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name `<network_gw_changes_metric>` --metric-transformations`

metricName= `<network_gw_changes_metric>` ,metricNamespace='CISBenchmark',metricValue=1 --filter-pattern '{($.eventName = CreateCustomerGateway) || ($.eventName = DeleteCustomerGateway) || ($.eventName = AttachInternetGateway) || ($.eventName = CreateInternetGateway) || ($.eventName = DeleteInternetGateway) || ($.eventName = DetachInternetGateway) }' ```
Note: You can choose your own metricName and metricNamespace strings. Using the
same metricNamespace for all Foundations Benchmark metrics will group them together. 2. Create an SNS topic that the alarm will notify
```
aws sns create-topic --name <sns_topic_name>
```

Note: you can execute this command once and then re-use the same topic for all
monitoring alarms. 3. Create an SNS subscription to the topic created in step 2
```
aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --notification-endpoint <sns_subscription_endpoints>
```

Note: you can execute this command once and then re-use the SNS subscription for all
monitoring alarms. 4. Create an alarm that is associated with the CloudWatch Logs Metric Filter created in step 1 and an SNS topic created in step 2
```
aws cloudwatch put-metric-alarm --alarm-name `<network_gw_changes_alarm>` --metric-name `<network_gw_changes_metric>` --statistic Su
```

Control: Ensure route table changes are monitored

Sections: 4.13 19 0

Description: Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs, or an external Security information and event management (SIEM) environment, and establishing corresponding metric filters and alarms. Routing tables are used to route network traffic between subnets and to network gateways. It is recommended that a metric filter and alarm be established for changes to route tables.

Remediation Steps:
If you are using CloudTrails and CloudWatch, perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for route table changes and the <cloudtrail_log_group_name> taken from audit step 1.

   ```
   aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name `<route_table_changes_metric>` --metric-transformationsmetricName= `<route_table\_changes_metric>`,metricNamespace='CISBenchmark',metricValue=1 --filter-pattern '{($.eventName = CreateRoute) || ($.eventName = CreateRouteTable) ||($.eventName = ReplaceRoute) || ($.eventName = ReplaceRouteTableAssociation)|| ($.eventName = DeleteRouteTable) || ($.eventName = DeleteRoute) ||($.eventName = DisassociateRouteTable) }'
   ```

   Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.
2. Create an SNS topic that the alarm will notify

   ```
   aws sns create-topic --name <sns_topic_name>
   ```

   Note: you can execute this command once and then re-use the same topic for all monitoring alarms.
3. Create an SNS subscription to the topic created in step 2

   ```
   aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --notification-endpoint <sns_subscription_endpoints>
   ```

   Note: you can execute this command once and then re-use the SNS subscription for all monitoring alarms.
4. Create an alarm that is associated with the CloudWatch Logs Metric Filter created in step 1 and an SNS topic created in step 2

   ```
   aws cloudwatch put-metric-alarm --alarm-name `<route_table_changes_alarm>` --metric-name `<route_table_changes_metric>` --statistic Sum --period 300 --threshold 1 --comparison-operator GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace 'CISBenchmark' --alarm-actions <sns_topic_arn>
   ```

Control: Ensure VPC changes are monitored

Sections: 4.14 19 0

Description: Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs, or an external Security information and event management (SIEM) environment, and establishing corresponding metric filters and alarms. It is possible to have more than 1 VPC within an account, in addition it is also possible to create a peer connection between 2 VPCs enabling network traffic to route between VPCs. It is recommended that a metric filter and alarm be established for changes made to VPCs.

Remediation Steps:
If you are using CloudTrails and CloudWatch, perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for VPC changes and the <cloudtrail_log_group_name> taken from audit step 1.

   ```
   aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name `<vpc_changes_metric>` --metric-transformations metricName=`<vpc_changes_metric>` ,metricNamespace='CISBenchmark',metricValue=1 --filter-pattern '{ ($.eventName = CreateVpc) || ($.eventName = DeleteVpc) ||($.eventName = ModifyVpcAttribute) || ($.eventName =AcceptVpcPeeringConnection) || ($.eventName = CreateVpcPeeringConnection) ||($.eventName = DeleteVpcPeeringConnection) || ($.eventName = RejectVpcPeeringConnection) || ($.eventName = AttachClassicLinkVpc) ||($.eventName = DetachClassicLinkVpc) || ($.eventName = DisableVpcClassicLink)|| ($.eventName = EnableVpcClassicLink) }'
   ```

   Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.
2. Create an SNS topic that the alarm will notify

   ```
   aws sns create-topic --name <sns_topic_name>
   ```

   Note: you can execute this command once and then re-use the same topic for all monitoring alarms.
3. Create an SNS subscription to the topic created in step 2

   ```
   aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --notification-endpoint <sns_subscription_endpoints>
   ```

   Note: you can execute this command once and then re-use the SNS subscription for all monitoring alarms.
4. Create an alarm that is associated with the CloudWatch Logs Metric Filter created in step 1 and an SNS topic created in step 2

   ```
   aws cloudwatch put-metric-alarm --alarm-name `<vpc_changes_alarm>` --metric-name `<vpc_changes_metric>` --statistic Sum --period 300 --threshold 1 --comparison-operator GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace 'CISBenchmark' --alarm-actions <sns_topic_arn>
   ```

Control: Ensure AWS Organizations changes are monitored

Sections: 4.15 19 0

Description: Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs, and establishing corresponding metric filters and alarms. It is recommended that a metric filter and alarm be established for AWS Organizations changes made in the master AWS Account.

Remediation Steps:
If you are using CloudTrails and CloudWatch, perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for AWS Organizations changes and the <cloudtrail_log_group_name> taken from audit Step 1:

```
aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name `<organizations_changes>` --metric-transformations metricName=`<organizations_changes>` ,metricNamespace='CISBenchmark',metricValue=1 --filter-pattern '{ ($.eventSource = organizations.amazonaws.com) &&(($.eventName = "AcceptHandshake") || ($.eventName = "AttachPolicy") || ($.eventName = "CreateAccount") || ($.eventName = "CreateOrganizationalUnit")|| ($.eventName = "CreatePolicy") || ($.eventName = "DeclineHandshake") ||($.eventName = "DeleteOrganization") || ($.eventName ="DeleteOrganizationalUnit") || ($.eventName = "DeletePolicy") ||
```

($.eventName = "DetachPolicy") || ($.eventName = "DisablePolicyType") ||($.eventName = "EnablePolicyType") || ($.eventName ="InviteAccountToOrganization") || ($.eventName = "LeaveOrganization") ||($.eventName = "MoveAccount") || ($.eventName ="RemoveAccountFromOrganization") || ($.eventName = "UpdatePolicy") ||($.eventName = "UpdateOrganizationalUnit")) }' ```
Note: You can choose your own metricName and metricNamespace strings. Using the

same metricNamespace for all Foundations Benchmark metrics will group them together. 2. Create an SNS topic that the alarm will notify:

```
aws sns create-topic --name <sns_topic_name>
```

Note: you can execute this command once and then re-use the same topic for all

monitoring alarms. 3. Create an SNS subscription to the topic created in step 2:

```
aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --notification-endpoint <sns_subscription_endpoints>
```

Note: you can execute this command once and then re-use the SNS subscription for all

monitoring alarms. 4. Create an alarm that is associated with the CloudWatch Logs Metric Filter created in step 1 and an SNS topic created in step 2:

```
aws cloudwatch put-metric-alarm --alarm-name `<organizations_changes>` --metric-name `<organizations_changes>` --statistic Sum --per
```

Control: Ensure AWS Security Hub is enabled

Sections: 4.16 284 39

Description: Security Hub collects security data from across AWS accounts, services, and supported third-party partner products and helps you analyze your security trends and identify the highest priority security issues. When you enable Security Hub, it begins to consume, aggregate, organize, and prioritize findings from AWS services that you have enabled, such as Amazon GuardDuty, Amazon Inspector, and Amazon Macie. You can also enable integrations with AWS partner security products.

Remediation Steps:
To grant the permissions required to enable Security Hub, attach the Security Hub managed policy AWSSecurityHubFullAccess to an IAM user, group, or role.

Enabling Security Hub

From Console:

1. Use the credentials of the IAM identity to sign in to the Security Hub console.
2. When you open the Security Hub console for the first time, choose Enable AWS Security Hub.
3. On the welcome page, Security standards list the security standards that Security Hub supports.
4. Choose Enable Security Hub.

From Command Line:

1. Run the enable-security-hub command. To enable the default standards, include --enable-default-standards.

   ```
   aws securityhub enable-security-hub –enable-default-standards
   ```
2. To enable the security hub without the default standards, include –no-enabledefault-standards.

   ```
   aws securityhub enable-security-hub –no-enable-default-standards
   ```

Control: Ensure no Network ACLs allow ingress from 0.0.0.0/0 to remote server administration ports

Sections: 5.1 445 0

Description: The Network Access Control List (NACL) function provide stateless filtering of ingress and egress network traffic to AWS resources. It is recommended that no NACL allows unrestricted ingress access to remote server administration ports, such as SSH to port 22 and RDP to port 3389, using either the TDP (6), UDP (17) or ALL (-1) protocols

Remediation Steps:
From Console:

Perform the following:

1. Login to the AWS Management Console at https://console.aws.amazon.com/vpc/home
2. In the left pane, click Network ACLs
3. For each network ACL to remediate, perform the following:
   - Select the network ACL
   - Click the Inbound Rules tab
   - Click Edit inbound rules
   - Either A) update the Source field to a range other than 0.0.0.0/0, or, B) Click Delete to remove the offending inbound rule
   - Click Save

Control: Ensure no security groups allow ingress from 0.0.0.0/0 to remote server administration ports

Sections: 5.2 223 1043

Description: Security groups provide stateful filtering of ingress and egress network traffic to AWS resources. It is recommended that no security group allows unrestricted ingress access to remote server administration ports, such as SSH to port 22 and RDP to port 3389, using either the TDP (6), UDP (17) or ALL (-1) protocols

Remediation Steps:
Perform the following to implement the prescribed state:

1. Login to the AWS Management Console at https://console.aws.amazon.com/vpc/home
2. In the left pane, click Security Groups
3. For each security group, perform the following:
4. Select the security group
5. Click the Inbound Rules tab
6. Click the Edit inbound rules button
7. Identify the rules to be edited or removed
8. Either A) update the Source field to a range other than 0.0.0.0/0, or, B) Click Delete to remove the offending inbound rule
9. Click Save rules

Control: Ensure no security groups allow ingress from ::/0 to remote server administration ports

Sections: 5.3 10 1256

Description: Security groups provide stateful filtering of ingress and egress network traffic to AWS resources. It is recommended that no security group allows unrestricted ingress access to remote server administration ports, such as SSH to port 22 and RDP to port 3389.

Remediation Steps:
Perform the following to implement the prescribed state:

1. Login to the AWS Management Console at https://console.aws.amazon.com/vpc/home
2. In the left pane, click Security Groups
3. For each security group, perform the following:
4. Select the security group
5. Click the Inbound Rules tab
6. Click the Edit inbound rules button
7. Identify the rules to be edited or removed
8. Either A) update the Source field to a range other than ::/0, or, B) Click Delete to remove the offending inbound rule
9. Click Save rules

Control: Ensure the default security group of every VPC restricts all traffic

Sections: 5.4 423 1

Description: A VPC comes with a default security group whose initial settings deny all inbound traffic, allow all outbound traffic, and allow all traffic between instances assigned to the security group. If you don't specify a security group when you launch an instance, the instance is automatically assigned to this default security group. Security groups provide stateful filtering of ingress/egress network traffic to AWS resources. It is recommended that the default security group restrict all traffic. The default VPC in every region should have its default security group updated to comply. Any newly created VPCs will automatically contain a default security group that will need remediation to comply with this recommendation. NOTE: When implementing this recommendation, VPC flow logging is invaluable in determining the least privilege port access required by systems to work properly because it can log all packet acceptances and rejections occurring under the current security groups. This dramatically reduces the primary barrier to least privilege engineering - discovering the minimum ports required by systems in the environment. Even if the VPC flow logging recommendation in this benchmark is not adopted as a permanent security measure, it should be used during any period of discovery and engineering for least privileged security groups.

Remediation Steps:
Security Group Members Perform the following to implement the prescribed state:

1. Identify AWS resources that exist within the default security group
2. Create a set of least privilege security groups for those resources
3. Place the resources in those security groups
4. Remove the resources noted in #1 from the default security group

Security Group State

1. Login to the AWS Management Console at https://console.aws.amazon.com/vpc/home
2. Repeat the next steps for all VPCs - including the default VPC in each AWS region:
3. In the left pane, click Security Groups
4. For each default security group, perform the following:
5. Select the default security group
6. Click the Inbound Rules tab
7. Remove any inbound rules
8. Click the Outbound Rules tab
9. Remove any Outbound rules

Recommended: IAM groups allow you to edit the "name" field. After remediating default groups rules for all VPCs in all regions, edit this field to add text similar to "DO NOT USE. DO NOT ADD RULES"

Control: Ensure that EC2 Metadata Service only allows IMDSv2

Sections: 5.6 3461 244

Description: When enabling the Metadata Service on AWS EC2 instances, users have the option of using either Instance Metadata Service Version 1 (IMDSv1; a request/response method) or Instance Metadata Service Version 2 (IMDSv2; a session-oriented method).

Remediation Steps:
From Console:

1. Login to AWS Management Console and open the Amazon EC2 console using https://console.aws.amazon.com/ec2/
2. Under the Instances menu, select Instances.
3. For each Instance, select the instance, then choose Actions > Modify instance metadata options.
4. If the Instance metadata service is enabled, set IMDSv2 to Required.

From Command Line:

```
aws ec2 modify-instance-metadata-options --instance-id <instance_id> --httptokens required
```