CIS Benchmarks

CIS Amazon Web Services Foundations Benchmark

v1.3.0 - 08-07-2020

Terms of Use

Please see the below link for our current terms of use:

https://www.cisecurity.org/cis-securesuite/cis-securesuite-membership-terms-of-use/

Table of Contents

Overview

This document provides prescriptive guidance for configuring security options for a subset of Amazon Web Services with an emphasis on foundational, testable, and architecture agnostic settings. Specific Amazon Web Services in scope for this document include:

- AWS Identity and Access Management (IAM)
- AWS Config
- AWS CloudTrail
- AWS CloudWatch
- AWS Simple Notification Service (SNS)
- AWS Simple Storage Service (S3)
- AWS VPC (Default)

To obtain the latest version of this guide, please visit http://benchmarks.cisecurity.org. If you have questions, comments, or have identified ways to improve this guide, please write us at BenchmarkInfo@cisecurity.org.

Intended Audience

This document is intended for system and application administrators, security specialists, auditors, help desk, platform deployment, and/or DevOps personnel who plan to develop, deploy, assess, or secure solutions in Amazon Web Services.

Consensus Guidance

This benchmark was created using a consensus review process comprised of subject matter experts. Consensus participants provide perspective from a diverse set of backgrounds including consulting, software development, audit and compliance, security research, operations, government, and legal.

Each CIS benchmark undergoes two phases of consensus review. The first phase occurs during initial benchmark development. During this phase, subject matter experts convene to discuss, create, and test working drafts of the benchmark. This discussion occurs until consensus has been reached on benchmark recommendations. The second phase begins after the benchmark has been published. During this phase, all feedback provided by the Internet community is reviewed by the consensus team for incorporation in the benchmark. If you are interested in participating in the consensus process, please visit https://workbench.cisecurity.org/.

Typographical Conventions

The following typographical conventions are used throughout this guide:

Convention	Meaning
`Stylized Monospace font`	Used for blocks of code, command, and script examples. Text should be interpreted exactly as presented.
Monospace font	Used for inline code, commands, or examples. Text should be interpreted exactly as presented.
<italic font in brackets>	Italic texts set in angle brackets denote a variable requiring substitution for a real value.
Italic font	Used to denote the title of a book, article, or other publication.
Note	Additional information or caveats

Assessment Status

An assessment status is included for every recommendation. The assessment status indicates whether the given recommendation can be automated or requires manual steps to implement. Both statuses are equally important and are determined and supported as defined below:

Automated

Represents recommendations for which assessment of a technical control can be fully automated and validated to a pass/fail state. Recommendations will include the necessary information to implement automation.

Manual

Represents recommendations for which assessment of a technical control cannot be fully automated and requires all or some manual steps to validate that the configured state is set as expected. The expected state can vary depending on the environment.

Profile Definitions

The following configuration profiles are defined by this Benchmark:

- **Level 1**

  Items in this profile intend to:

  - be practical and prudent;
  - provide a clear security benefit; and
  - not inhibit the utility of the technology beyond acceptable means.

- **Level 2**

  This profile extends the "Level 1" profile. Items in this profile exhibit one or more of the following characteristics:

  - are intended for environments or use cases where security is paramount
  - acts as defense in depth measure
  - may negatively inhibit the utility or performance of the technology.

Acknowledgements

This benchmark exemplifies the great things a community of users, vendors, and subject matter experts can accomplish through consensus collaboration. The CIS community thanks the entire consensus team with special recognition to the following individuals who contributed greatly to the creation of this guide:

Contributor
Cindy Spiess, Adobe
Blake Frantz
Gavin Fitzpatrick
Amol Pathak
James Covington
Rob Witoff
John Martinez
Tim Sandage
Mike De Libero, MDE Development, Inc.
Adam Montville
Darwin Sanoy
Gregory Frascadore
Ionut Dragoi
John Robel
Brent Harrison
Mike Wicks
Aditi Sahasrabudhe
Jeremy Phillips
Ankit Rao

Editor
Iben Rodriguez
Parag Patil
Pradeep R B
Gregory Carpenter
Maril Vernon
Paul Campbell

Recommendations

1 Identity and Access Management

This section contains recommendations for configuring identity and access management related options.

1.1 Maintain current contact details (Manual)

Profile Applicability:

- Level 1

Description:

Ensure contact email and telephone details for AWS accounts are current and map to more than one individual in your organization.

An AWS account supports a number of contact details, and AWS will use these to contact the account owner if activity judged to be in breach of Acceptable Use Policy or indicative of likely security compromise is observed by the AWS Abuse team. Contact details should not be for a single individual, as circumstances may arise where that individual is unavailable. Email contact details should point to a mail alias which forwards email to multiple individuals within the organization; where feasible, phone contact details should point to a PABX hunt group or other call-forwarding system.

Rationale:

If an AWS account is observed to be behaving in a prohibited or suspicious manner, AWS will attempt to contact the account owner by email and phone using the contact details listed. If this is unsuccessful and the account behavior needs urgent mitigation, proactive measures may be taken, including throttling of traffic between the account exhibiting suspicious behavior and the AWS API endpoints and the Internet. This will result in impaired service to and from the account in question, so it is in both the customers' and AWS' best interests that prompt contact can be established. This is best achieved by setting AWS account contact details to point to resources which have multiple individuals as recipients, such as email aliases and PABX hunt groups.

Audit:

This activity can only be performed via the AWS Console, with a user who has permission to read and write Billing information (aws-portal:*Billing)

1. Sign in to the AWS Management Console and open the Billing and Cost Management console at https://console.aws.amazon.com/billing/home#/.
2. On the navigation bar, choose your account name, and then choose My Account.
3. On the Account Settings page, review and verify the current details.
4. Under Contact Information, review and verify the current details.

Remediation:

This activity can only be performed via the AWS Console, with a user who has permission to read and write Billing information (aws-portal:*Billing).

1. Sign in to the AWS Management Console and open the Billing and Cost Management console at https://console.aws.amazon.com/billing/home#/.
2. On the navigation bar, choose your account name, and then choose My Account.
3. On the Account Settings page, next to Account Settings, choose Edit.
4. Next to the field that you need to update, choose Edit.
5. After you have entered your changes, choose Save changes.
6. After you have made your changes, choose Done.
7. To edit your contact information, under Contact Information, choose Edit.
8. For the fields that you want to change, type your updated information, and then choose Update.

References:

1. https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/manage-account-payment.html#contact-info

CIS Controls:

Version 7

6.3 Enable Detailed Logging

Enable system logging to include detailed information such as an event source, date, user, timestamp, source addresses, destination addresses, and other useful elements.

1.2 Ensure security contact information is registered (Manual)

Profile Applicability:

- Level 1

Description:

AWS provides customers with the option of specifying the contact information for account's security team. It is recommended that this information be provided.

Rationale:

Specifying security-specific contact information will help ensure that security advisories sent by AWS reach the team in your organization that is best equipped to respond to them.

Audit:

Perform the following to determine if security contact information is present:
From Console:

1. Click on your account name at the top right corner of the console
2. From the drop-down menu Click `My Account`
3. Scroll down to the `Alternate Contacts` section
4. Ensure contact information is specified in the `Security` section

Remediation:

Perform the following to establish security contact information:
From Console:

1. Click on your account name at the top right corner of the console.
2. From the drop-down menu Click `My Account`
3. Scroll down to the `Alternate Contacts` section
4. Enter contact information in the `Security` section

Note: Consider specifying an internal email distribution list to ensure emails are regularly monitored by more than one individual.

References:

1. CCE-79200-2

CIS Controls:

Version 7

19 Incident Response and Management
Incident Response and Management

19.2 Assign Job Titles and Duties for Incident Response
Assign job titles and duties for handling computer and network incidents to specific individuals and ensure tracking and documentation throughout the incident through resolution.

1.3 Ensure security questions are registered in the AWS account (Manual)

Profile Applicability:

- Level 1

Description:

The AWS support portal allows account owners to establish security questions that can be used to authenticate individuals calling AWS customer service for support. It is recommended that security questions be established.

Rationale:

When creating a new AWS account, a default super user is automatically created. This account is referred to as the "root user" account. It is recommended that the use of this account be limited and highly controlled. During events in which the Root password is no longer accessible or the MFA token associated with root is lost/destroyed it is possible, through authentication using secret questions and associated answers, to recover root user login access.

Audit:

From Console:

1. Login to the AWS account as the root user
2. On the top right you will see the *<Root_Account_Name>*
3. Click on the *<Root_Account_Name>*
4. From the drop-down menu Click `My Account`
5. In the `Configure Security Challenge Questions` section on the `Personal Information` page, configure three security challenge questions.
6. Click `Save questions`.

Remediation:

From Console:

1. Login to the AWS Account as the root user
2. Click on the *<Root_Account_Name>* from the top right of the console
3. From the drop-down menu Click *My Account*
4. Scroll down to the `Configure Security Questions` section
5. Click on `Edit`
6. Click on each `Question`

- From the drop-down select an appropriate question
- Click on the `Answer` section
- Enter an appropriate answer
    - Follow process for all 3 questions

7. Click `Update` when complete
8. Place Questions and Answers and place in a secure physical location

CIS Controls:

Version 7

16 Account Monitoring and Control

Account Monitoring and Control

1.4 Ensure no root user account access key exists (Automated)

Profile Applicability:

- Level 1

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 removed.

Rationale:

Removing access keys associated with the root user account limits vectors by which the account can be compromised. Additionally, removing the root access keys encourages the creation and use of role based accounts that are least privileged.

Audit:

Perform the following to determine if the root user account has access keys:
From Console:

1. Login to the AWS Management Console
2. Click `Services`
3. Click `IAM`
4. Click on `Credential Report`
5. This will download an `.xls` file which contains credential usage for all IAM users within an AWS Account - open this file
6. For the `<root_account>` user, ensure the `access_key_1_active` and `access_key_2_active` fields are set to `FALSE` .

From Command Line:
Run the following command:

```
aws iam get-account-summary | grep "AccountAccessKeysPresent"
```

If no root access keys exist the output will show "AccountAccessKeysPresent": 0,.
If the output shows a "1" than root keys exist, refer to the remediation procedure below.

Remediation:

Perform the following to delete or disable 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_Name>* 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
    1. Click on `Make Inactive` - (Temporarily disable Key - may be needed again)
    2. Click `Delete` - (Deleted keys cannot be recovered)

References:

1. http://docs.aws.amazon.com/general/latest/gr/aws-access-keys-best-practices.html
2. http://docs.aws.amazon.com/general/latest/gr/managing-aws-access-keys.html
3. http://docs.aws.amazon.com/IAM/latest/APIReference/API_GetAccountSummary.html
4. CCE-78910-7
5. https://aws.amazon.com/blogs/security/an-easier-way-to-determine-the-presence-of-aws-account-access-keys/

Additional Information:

IAM User account "root" for us-gov cloud regions is not enabled by default. However, on request to AWS support enables root access only through access-keys (CLI, API methods) for us-gov cloud region.

CIS Controls:

Version 7

4.3 Ensure the Use of Dedicated Administrative Accounts

Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.

1.5 Ensure MFA is enabled for the "root user" account (Automated)

Profile Applicability:

- Level 1

Description:

The root user account is the most privileged user in an AWS account. Multi-factor Authentication (MFA) adds an extra layer of protection on top of a username and password. With MFA enabled, when a user signs in to an AWS website, they will be prompted for their username and password as well as for an authentication code from their AWS MFA device.

Note: When virtual MFA is used for root accounts, it is recommended that the device used is NOT a personal device, but rather a dedicated mobile device (tablet or phone) that is managed to be kept charged and secured independent of any individual personal devices. ("non-personal virtual MFA") This lessens the risks of losing access to the MFA due to device loss, device trade-in or if the individual owning the device is no longer employed at the company.

Rationale:

Enabling MFA provides increased security for console access as it requires the authenticating principal to possess a device that emits a time-sensitive key and have knowledge of a credential.

Audit:

Perform the following to determine if the root user account has MFA setup:
From Command Line:

1. Run the following command:

```
aws iam get-account-summary | grep "AccountMFAEnabled"
```

2. Ensure the AccountMFAEnabled property is set to 1

Remediation:

Perform the following to establish 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 root AWS 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 virtual MFA` device and then choose `Next Step`.
5. 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.) 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.
In the Manage MFA Device wizard, in the Authentication 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 one-time password into the Authentication Code 2 box. Choose Active Virtual MFA.

References:

1. CCE-78911-5
2. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_root-user.html#id_root-user_manage_mfa
3. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_enable_virtual.html#enable-virt-mfa-for-root

Additional Information:

IAM User account "root" for us-gov cloud regions does not have console access. This control is not applicable for us-gov cloud regions.

CIS Controls:

Version 7

4.5 Use Multifactor Authentication For All Administrative Access

Use multi-factor authentication and encrypted channels for all administrative account access.

1.6 Ensure hardware MFA is enabled for the "root user" account (Automated)

Profile Applicability:

- Level 2

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.

Rationale:

A hardware MFA has a smaller attack surface than a virtual MFA. For example, a hardware MFA does not suffer the attack surface introduced by the mobile smartphone on which a virtual MFA resides.

Note: Using hardware MFA for many, many AWS accounts may create a logistical device management issue. If this is the case, consider implementing this Level 2 recommendation selectively to the highest security AWS accounts and the Level 1 recommendation applied to the remaining accounts.

Audit:

Perform the following to determine if the root user account has a hardware MFA setup:

1. Run the following command to determine if the root account has MFA setup:

```
aws iam get-account-summary | grep "AccountMFAEnabled"
```

The `AccountMFAEnabled` property is set to `1` will ensure that the root user account has MFA (Virtual or Hardware) Enabled.
If `AccountMFAEnabled` property is set to `0` the account is not compliant with this recommendation.

2. If `AccountMFAEnabled` property is set to `1`, determine root account has Hardware MFA enabled.
   Run the following command to list all virtual MFA devices:

```
aws iam list-virtual-mfa-devices
```

If the output contains one MFA with the following Serial Number, it means the MFA is virtual, not hardware and the account is not compliant with this recommendation:
`"SerialNumber": "arn:aws:iam::_<aws_account_number>_:mfa/root-account-mfa-device"`

Remediation:

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.

References:

1. CCE-78911-5
2. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_enable_virtual.html
3. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_enable_physical.html#enable-hw-mfa-for-root

Additional Information:

IAM User account "root" for us-gov cloud regions does not have console access. This control is not applicable for us-gov cloud regions.

CIS Controls:

Version 7

4.5 Use Multifactor Authentication For All Administrative Access

Use multi-factor authentication and encrypted channels for all administrative account access.

1.7 Eliminate use of the root user for administrative and daily tasks (Automated)

Profile Applicability:

- Level 1

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.

Rationale:

The *root user* has unrestricted access to and control over all account resources. Use of it is inconsistent with the principles of least privilege and separation of duties, and can lead to unnecessary harm due to error or account compromise.

Audit:

From Console:

1. Login to the AWS Management Console at `https://console.aws.amazon.com/iam/`
2. In the left pane, click `Credential Report`
3. Click on `Download Report`
4. Open of Save the file locally
5. Locate the `<root account>` under the user column
6. Review `password_last_used, access_key_1_last_used_date, access_key_2_last_used_date` to determine when the *root user* was last used.

From Command Line:
Run the following CLI commands to provide a credential report for determining the last time the *root user* was used:

```
aws iam generate-credential-report

aws iam get-credential-report --query 'Content' --output text | base64 -d |
cut -d, -f1,5,11,16 | grep -B1 '<root_account>'
```

Review `password_last_used`, `access_key_1_last_used_date`, `access_key_2_last_used_date` to determine when the *root user* was last used.

Note: There are a few conditions under which the use of the *root user* account is required. Please see the reference links for all of the tasks that require use of the *root user*.

Remediation:

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.

References:

1. https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html
2. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_root-user.html
3. https://docs.aws.amazon.com/general/latest/gr/aws_tasks-that-require-root.html

Additional Information:

The *root user* for us-gov cloud regions is not enabled by default. However, on request to AWS support, they can enable the *root user* and grant access only through access-keys (CLI, API methods) for us-gov cloud region. If the *root user* for us-gov cloud regions is enabled, this recommendation is applicable.

Monitoring usage of the *root user* can be accomplished by implementing recommendation 3.3 Ensure a log metric filter and alarm exist for usage of the *root user*.

CIS Controls:

Version 7

4.3 Ensure the Use of Dedicated Administrative Accounts

Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.

1.8 Ensure IAM password policy requires minimum length of 14 or greater (Automated)

Profile Applicability:

- Level 1

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.

Rationale:

Setting a password complexity policy increases account resiliency against brute force login attempts.

Audit:

Perform the following to ensure the password policy is configured 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. Ensure "Minimum password length" is set to 14 or greater.

From Command Line:

```
aws iam get-account-password-policy
```

Ensure the output of the above command includes "MinimumPasswordLength": 14 (or higher)

Remediation:

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.

References:

1. CCE-78907-3
2. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_passwords_account-policy.html
3. https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#configure-strong-password-policy

CIS Controls:

Version 7

16 Account Monitoring and Control
Account Monitoring and Control

1.9 Ensure IAM password policy prevents password reuse (Automated)

Profile Applicability:

- Level 1

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.

Rationale:

Preventing password reuse increases account resiliency against brute force login attempts.

Audit:

Perform the following to ensure the password policy is configured 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. Ensure "Prevent password reuse" is checked
5. Ensure "Number of passwords to remember" is set to 24

From Command Line:

```
aws iam get-account-password-policy
```

Ensure the output of the above command includes "PasswordReusePrevention": 24

Remediation:

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.

References:

1. CCE-78908-1
2. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_passwords_account-policy.html
3. https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#configure-strong-password-policy

CIS Controls:

Version 7

4.4 Use Unique Passwords

Where multi-factor authentication is not supported (such as local administrator, root, or service accounts), accounts will use passwords that are unique to that system.

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

Profile Applicability:

- Level 1

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.

Rationale:

Enabling MFA provides increased security for console access as it requires the authenticating principal to possess a device that displays a time-sensitive key and have knowledge of a credential.

Audit:

Perform the following to determine if a MFA device is enabled for all IAM users having a console password:

From Console:

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left pane, select `Users`
3. If the `MFA Device` or `Password` columns are not visible in the table, click the gear icon at the upper right corner of the table and ensure a checkmark is next to both, then click `Close` .
4. Ensure that for each user where the `Password` column shows a password age, the `MFA Device` column shows `Virtual`, `U2F Security Key`, `Hardware`, or SMS.

From Command Line:

1. Run the following command (OSX/Linux/UNIX) to generate a list of all IAM users along with their password and MFA status:

```
  aws iam generate-credential-report

  aws iam get-credential-report --query 'Content' --output text | base64 -d |
cut -d, -f1,4,8
```

2. The output of this command will produce a table similar to the following:

```
user,password_enabled,mfa_active
elise,false,false
brandon,true,true
rakesh,false,false
helene,false,false
paras,true,true
anitha,false,false
```

3. For any column having `password_enabled` set to `true` , ensure `mfa_active` is also set to `true`.

Remediation:

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 navigation pane, choose 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.) 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 one-time password into the MFA Code 2 box. Choose Assign MFA.

Forced IAM User Self-Service Remediation

Amazon has published a pattern that forces users to self-service setup MFA before they have access to their complete permissions set. Until they complete this step, they cannot access their full permissions. This pattern can be used on new AWS accounts. It can also be used on existing accounts - it is recommended users are given instructions and a grace period to accomplish MFA enrollment before active enforcement on existing AWS accounts. How to Delegate Management of Multi-Factor Authentication to AWS IAM Users

References:

1. https://tools.ietf.org/html/rfc6238
2. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa.html
3. https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#enable-mfa-for-privileged-users
4. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_enable_virtual.html
5. CCE-78901-6

CIS Controls:

Version 7

4.5 Use Multifactor Authentication For All Administrative Access

Use multi-factor authentication and encrypted channels for all administrative account access.

1.11 Do not setup access keys during initial user setup for all IAM users that have a console password (Manual)

Profile Applicability:

- Level 1

Description:

AWS console defaults to no check boxes selected when creating a new IAM user. When cerating 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.

Rationale:

Requiring the additional steps be taken by the user for programmatic access after their profile has been created will give a stronger indication of intent that access keys are [a] necessary for their work and [b] once the access key is established on an account that the keys may be in use somewhere in the organization.

Note: Even if it is known the user will need access keys, require them to create the keys themselves or put in a support ticket to have them created as a separate step from user creation.

Audit:

Perform the following to determine if access keys were created upon user creation and are being used and rotated as prescribed:
From Console:

1. Login to the AWS Management Console
2. Click `Services`
3. Click `IAM`
4. Click on a User where column `Password age` and `Access key age` is not set to `None`
5. Click on `Security credentials` Tab
6. Compare the user 'Creation time`to the Access Key`Created` date.
7. For any that match, the key was created during initial user setup.

- Keys that were created at the same time as the user profile and do not have a last used date should be deleted. Refer to the remediation below.

From Command Line:

1. Run the following command (OSX/Linux/UNIX) to generate a list of all IAM users along with their access keys utilization:

```
  aws iam generate-credential-report

  aws iam get-credential-report --query 'Content' --output text | base64 -d |
cut -d, -f1,4,9,11,14,16
```

2. The output of this command will produce a table similar to the following:

```
user,password_enabled,access_key_1_active,access_key_1_last_used_date,access_
key_2_active,access_key_2_last_used_date
  elise,false,true,2015-04-16T15:14:00+00:00,false,N/A
  brandon,true,true,N/A,false,N/A
  rakesh,false,false,N/A,false,N/A
  helene,false,true,2015-11-18T17:47:00+00:00,false,N/A
  paras,true,true,2016-08-28T12:04:00+00:00,true,2016-03-04T10:11:00+00:00
  anitha,true,true,2016-06-08T11:43:00+00:00,true,N/A
```

3. For any user having `password_enabled` set to `true` AND `access_key_last_used_date` set to `N/A` refer to the remediation below.

Remediation:

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>
```

References:

1. https://docs.aws.amazon.com/cli/latest/reference/iam/delete-access-key.html
2. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_create.html

Additional Information:

Credential report does not appear to contain "Key Creation Date" - maybe a feature request to AWS?

CIS Controls:

Version 7

16 Account Monitoring and Control
Account Monitoring and Control

1.12 Ensure credentials unused for 90 days or greater are disabled (Automated)

Profile Applicability:

- Level 1

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 90 or greater days be deactivated or removed.

Rationale:

Disabling or removing unnecessary credentials will reduce the window of opportunity for credentials associated with a compromised or abandoned account to be used.

Audit:

Perform the following to determine if unused credentials exist:
From Console:

1. Login to the AWS Management Console
2. Click `Services`
3. Click `IAM`
4. Click on `Users`
5. Click the `Settings` (gear) icon.
6. Select `Console last sign-in`, `Access key last used`, and `Access Key Id`
7. Click on `Close`
8. Check and ensure that `Console last sign-in` is less than 90 days ago.

Note - `Never` means the user has never logged in.

9. Check and ensure that `Access key age` is less than 90 days and that `Access key last used` does not say `None`

If the user hasn't signed into the Console in the last 90 days or Access keys are over 90 days old refer to the remediation.
From Command Line:
Download Credential Report:

1. Run the following commands:

```
  aws iam generate-credential-report

  aws iam get-credential-report --query 'Content' --output text | base64 -d |
cut -d, -f1,4,5,6,9,10,11,14,15,16
```

Ensure unused credentials do not exist:

2. For each user having `password_enabled` set to `TRUE` , ensure `password_last_used_date` is less than `90` days ago.
    - When `password_enabled` is set to `TRUE` and `password_last_used` is set to `No_Information` , ensure `password_last_changed` is less than 90 days ago.

3. For each user having an `access_key_1_active` or `access_key_2_active` to `TRUE` , ensure the corresponding `access_key_n_last_used_date` is less than `90` days ago.
    - When a user having an `access_key_x_active` (where x is 1 or 2) to `TRUE` and corresponding access_key_x_last_used_date is set to `N/A', ensure` access_key_x_last_rotated` is less than 90 days ago.

Remediation:

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 90 days
7. Click `Security credentials`
8. In section `Sign-in credentials`, `Console paassword` 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 90 days old and that have been used and

    - Click on `Make Inactive`

7. Select any access keys that are over 90 days old and that have not been used and

    - Click the X to `Delete`

References:

1. CCE-78900-8
2. https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#remove-credentials
3. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_finding-unused.html
4. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_passwords_admin-change-user.html
5. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html

CIS Controls:

Version 7

16.9 Disable Dormant Accounts
Automatically disable dormant accounts after a set period of inactivity.

1.13 Ensure there is only one active access key available for any single IAM user (Automated)

Profile Applicability:

- Level 1

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)

Rationale:

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. One of the best ways to protect your account is to not allow users to have multiple access keys.

Audit:

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. Under `Access Keys` section, in the Status column, check the current status for each access key associated with the IAM user. If the selected IAM user has more than one access key activated then the users access configuration does not adhere to security best practices and the risk of accidental exposures increases.

    - Repeat steps no. 3 – 5 for each IAM user in your AWS account.

From Command Line:

1. Run `list-users` command to list all IAM users within your account:

```
aws iam list-users --query "Users[*].UserName"
```

The command output should return an array that contains all your IAM user names.

2. Run `list-access-keys` command using the IAM user name list to return the current status of each access key associated with the selected IAM user:

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

The command output should expose the metadata `("Username", "AccessKeyId", "Status", "CreateDate")` for each access key on that user account.

3. Check the `Status` property value for each key returned to determine each keys current state. If the `Status` property value for more than one IAM access key is set to `Active`, the user access configuration does not adhere to this recommendation, refer to the remediation below.

- Repeat steps no. 2 and 3 for each IAM user in your AWS account.

Remediation:

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 – 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 IAM user

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 – 3 for each IAM user in your AWS account.

References:

1. https://docs.aws.amazon.com/general/latest/gr/aws-access-keys-best-practices.html
2. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html

CIS Controls:

Version 7

4 Controlled Use of Administrative Privileges
Controlled Use of Administrative Privileges

1.14 Ensure access keys are rotated every 90 days or less (Automated)

Profile Applicability:

- Level 1

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.

Rationale:

Rotating access keys will reduce the window of opportunity for an access key that is associated with a compromised or terminated account to be used.

Access keys should be rotated to ensure that data cannot be accessed with an old key which might have been lost, cracked, or stolen.

Audit:

Perform the following to determine if access keys are rotated as prescribed:
From Console:

1. Go to Management Console (https://console.aws.amazon.com/iam)
2. Click on Users
3. Click setting icon
4. Select “Console last sign-in”
5. Click Close
6. Ensure that “Access key age” is less than 90 days ago. note) "None" in the "Access key age" means the user has not used the access key.

From Command Line:

```
aws iam generate-credential-report
aws iam get-credential-report --query 'Content' --output text | base64 -d
```

The `access_key_1_last_rotated` field in this file notes The date and time, in ISO 8601 date-time format, when the user's access key was created or last changed. If the user does not have an active access key, the value in this field is N/A (not applicable).

Remediation:

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
```

References:

1. CCE-78902-4
2. https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#rotate-credentials
3. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_finding-unused.html
4. https://docs.aws.amazon.com/general/latest/gr/managing-aws-access-keys.html
5. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html

CIS Controls:

Version 7

16 Account Monitoring and Control
Account Monitoring and Control

1.15 Ensure IAM Users Receive Permissions Only Through Groups (Automated)

Profile Applicability:

- Level 1

Description:

IAM users are granted access to services, functions, and data through IAM policies. There are three 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.

Only the third implementation is recommended.

Rationale:

Assigning IAM policy only through groups unifies permissions management to a single, flexible layer consistent with organizational functional roles. By unifying permissions management, the likelihood of excessive permissions is reduced.

Audit:

Perform the following to determine if an inline policy is set or a policy is directly attached to users:

1. Run the following to get a list of IAM users:

```
aws iam list-users --query 'Users[*].UserName' --output text
```

2. For each user returned, run the following command to determine if any policies are attached to them:

```
aws iam list-attached-user-policies --user-name  <iam_user>
aws iam list-user-policies --user-name <iam_user>
```

3. If any policies are returned, the user has an inline policy or direct policy attachment.

Remediation:

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)

References:

1. http://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html
2. http://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html
3. CCE-78912-3

CIS Controls:

Version 7

16 Account Monitoring and Control
Account Monitoring and Control

1.16 Ensure IAM policies that allow full ":*" administrative privileges are not attached (Automated)*

Profile Applicability:

- Level 1

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.

Rationale:

It's more secure to start with a minimum set of permissions and grant additional permissions as necessary, rather than starting with permissions that are too lenient and then trying to tighten them later.

Providing full administrative privileges instead of restricting to the minimum set of permissions that the user is required to do exposes the resources to potentially unwanted actions.

IAM policies that have a statement with "Effect": "Allow" with "Action": "*" over "Resource": "*" should be removed.

Audit:

Perform the following to determine what policies are created:
From Command Line:

1. Run the following to get a list of IAM policies:

```
  aws iam list-policies --only-attached --output text
```

2. For each policy returned, run the following command to determine if any policies is allowing full administrative privileges on the account:

```
  aws iam get-policy-version  --policy-arn <policy_arn> --version-id
<version>
```

3. In output ensure policy should not have any Statement block with `"Effect": "Allow"` and `Action` set to `"*"` and `Resource` set to `"*"`

Remediation:

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>
```

References:

1. https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html
2. https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html

3. CCE-78912-3
4. https://docs.aws.amazon.com/cli/latest/reference/iam/index.html#cli-aws-iam

CIS Controls:

Version 7

4 Controlled Use of Administrative Privileges
Controlled Use of Administrative Privileges

1.17 Ensure a support role has been created to manage incidents with AWS Support (Automated)

Profile Applicability:

- Level 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 to allow authorized users to manage incidents with AWS Support.

Rationale:

By implementing least privilege for access control, an IAM Role will require an appropriate IAM Policy to allow Support Center Access in order to manage Incidents with AWS Support.

Impact:

All AWS Support plans include an unlimited number of account and billing support cases, with no long-term contracts. Support billing calculations are performed on a per-account basis for all plans. Enterprise Support plan customers have the option to include multiple enabled accounts in an aggregated monthly billing calculation. Monthly charges for the Business and Enterprise support plans are based on each month's AWS usage charges, subject to a monthly minimum, billed in advance.

Audit:

From Command Line:

1. List IAM policies, filter for the 'AWSSupportAccess' managed policy, and note the "Arn" element value:

```
aws iam list-policies --query "Policies[?PolicyName == 'AWSSupportAccess']"
```

2. Check if the 'AWSSupportAccess' policy is attached to any role:

```
aws iam list-entities-for-policy --policy-arn
arn:aws:iam::aws:policy/AWSSupportAccess
```

3. In Output, Ensure `PolicyRoles` does not return empty. 'Example: Example: PolicyRoles: [ ]'

If it returns empty refer to the remediation below.

Remediation:

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-policy-
document 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>
```

References:

1. https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html
2. https://aws.amazon.com/premiumsupport/pricing/
3. https://docs.aws.amazon.com/cli/latest/reference/iam/list-policies.html
4. https://docs.aws.amazon.com/cli/latest/reference/iam/attach-role-policy.html
5. https://docs.aws.amazon.com/cli/latest/reference/iam/list-entities-for-policy.html

Additional Information:

AWSSupportAccess policy is a global AWS resource. It has same ARN as `arn:aws:iam::aws:policy/AWSSupportAccess` for every account.

CIS Controls:

Version 7

14 Controlled Access Based on the Need to Know
Controlled Access Based on the Need to Know

1.18 Ensure IAM instance roles are used for AWS resource access from instances (Manual)

Profile Applicability:

- Level 2

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.

Rationale:

AWS IAM roles reduce the risks associated with sharing and rotating credentials that can be used outside of AWS itself. If credentials are compromised, they can be used from outside of the the AWS account they give access to. In contrast, in order to leverage role permissions an attacker would need to gain and maintain access to a specific instance to use the privileges associated with it.

Additionally, if credentials are encoded into compiled applications or other hard to change mechanisms, then they are even more unlikely to be properly rotated due to service disruption risks. As time goes on, credentials that cannot be rotated are more likely to be known by an increasing number of individuals who no longer work for the organization owning the credentials.

Audit:

Whether an Instance Is Associated With a Role
For instances that are known to perform AWS actions, ensure that they belong to an instance role that has the necessary permissions:

1. Login to AWS Console (with appropriate permissions to View Identity Access Management Account Settings)
2. Open the EC2 Dashboard and choose "Instances"
3. Click the EC2 instance that performs AWS actions, in the lower pane details find "IAM Role"
4. If the Role is blank, the instance is not assigned to one.
5. If the Role is filled in, it does not mean the instance might not *also* have credentials encoded on it for some activities.

Whether an Instance Contains Embedded Credentials
On the instance that is known to perform AWS actions, audit all scripts and environment variables to ensure that none of them contain AWS credentials.
Whether an Instance Application Contains Embedded Credentials
Applications that run on an instance may also have credentials embedded. This is a bad practice, but even worse if the source code is stored in a public code repository such as github. When an application contains credentials can be determined by eliminating all other sources of credentials and if the application can still access AWS resources - it likely contains embedded credentials. Another method is to examine all source code and configuration files of the application.

Remediation:

IAM roles can only be associated at the launch of an instance. To remediate an instance to add it to a role you must create a new instance.
If the instance has no external dependencies on its current private ip or public addresses are elastic IPs:

1. In AWS IAM create a new role. Assign a permissions policy if needed permissions are already known.
2. In the AWS console launch a new instance with identical settings to the existing instance, and ensure that the newly created role is selected.
3. Shutdown both the existing instance and the new instance.
4. Detach disks from both instances.
5. Attach the existing instance disks to the new instance.
6. Boot the new instance and you should have the same machine, but with the associated role.

Note: if your environment has dependencies on a dynamically assigned PRIVATE IP address you can create an AMI from the existing instance, destroy the old one and then when launching from the AMI, manually assign the previous private IP address.

Note: if your environment has dependencies on a dynamically assigned PUBLIC IP address there is not a way ensure the address is retained and assign an instance role. Dependencies on dynamically assigned public IP addresses are a bad practice and, if possible, you may wish to rebuild the instance with a new elastic IP address and make the investment to remediate affected systems while assigning the system to a role.

References:

1. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html
2. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html

CIS Controls:

Version 7

19 Incident Response and Management
Incident Response and Management

1.19 Ensure that all the expired SSL/TLS certificates stored in AWS IAM are removed (Automated)

Profile Applicability:

- Level 1

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.

Rationale:

Removing expired SSL/TLS certificates eliminates the risk that an invalid certificate will be deployed accidentally to a resource such as AWS Elastic Load Balancer (ELB), which can damage the credibility of the application/website behind the ELB. As a best practice, it is recommended to delete expired certificates.

Impact:

Deleting the certificate could have implications for your application If you are using a expired server certificate with Elastic Load Balancing, Cloudfront etc. . One has to make configurations at respective services to ensure there is no interruption in application.

Audit:

From Console:
Getting the certificates expiration information via AWS Management Console is not currently supported.
To request information about the SSL/TLS certificates stored in IAM via the AWS API use the Command Line Interface (CLI).

From Command Line:
Run list-server-certificates command to list all the IAM-stored server certificates:

```
aws iam list-server-certificates
```

The command output should return an array that contains all the SSL/TLS certificates currently stored in IAM and their metadata (name, ID, expiration date, etc):

```
{
    "ServerCertificateMetadataList": [
        {
            "ServerCertificateId": "EHDGFRW7EJFYTE88D",
            "ServerCertificateName": "MyServerCertificate",
            "Expiration": "2018-07-10T23:59:59Z",
            "Path": "/",
            "Arn": "arn:aws:iam::012345678910:server-
certificate/MySSLCertificate",
            "UploadDate": "2018-06-10T11:56:08Z"
        }
    ]
}
```

Verify the `ServerCertificateName` and `Expiration` parameter value (expiration date) for each SSL/TLS certificate returned by the list-server-certificates command and determine if there are any expired server certificates currently stored in AWS IAM. If so, use the AWS API to remove them.
If this command returns:

```
{ { "ServerCertificateMetadataList": [] }
```

This means that there are no expired certificates, It DOES NOT mean that no certificates exist.

Remediation:

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:
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.

Default Value:

By default, expired certificates won't get deleted.

References:

1. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_server-certs.html
2. https://docs.aws.amazon.com/cli/latest/reference/iam/delete-server-certificate.html

CIS Controls:

Version 7

13 Data Protection
Data Protection

1.20 Ensure that S3 Buckets are configured with 'Block public access (bucket settings)' (Automated)

Profile Applicability:

- Level 1

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 principle 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.

Rationale:

Amazon S3 `Block public access (bucket settings)` prevents the accidental or malicious public exposure of data contained within the respective bucket(s).

Amazon S3 `Block public access (account settings)` prevents the accidental or malicious public exposure of data contained within all buckets of the respective AWS account.

Whether blocking public access to all or some buckets is an organizational decision that should be based on data sensitivity, least privilege, and use case.

Impact:

When you apply Block Public Access settings to an account, the settings apply to all AWS Regions globally. The settings might not take effect in all Regions immediately or simultaneously, but they eventually propagate to all Regions.

Audit:

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. Ensure that block public access settings are set appropriately for this bucket
5. Repeat for all the buckets in your AWS account.

From Command Line:

1. List all of the S3 Buckets

```
aws s3 ls
```

2. Find the public access setting on that bucket

```
aws s3api get-public-access-block --bucket <name-of-the-bucket>
```

Output if Block Public access is enabled:

```
{
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    }
}
```

If the output reads `false` for the separate configuration settings then proceed to the remediation.

If utilizing Block Public Access (account settings)

From Console:

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. Ensure that block public access settings are set appropriately for your AWS account.

From Command Line:

To check Public access settings for this account status, run the following command,

```
aws s3control get-public-access-block --account-id <ACCT_ID> --region
<REGION_NAME>
```

Output if Block Public access is enabled:

```
{
    "PublicAccessBlockConfiguration": {
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "BlockPublicAcls": true,
```

```
        "RestrictPublicBuckets": true
    }
}
```

If the output reads `false` for the separate configuration settings then proceed to the remediation.

Remediation:

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 public access to true on that bucket

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

References:

1. https://docs.aws.amazon.com/AmazonS3/latest/user-guide/block-public-access-account.html

CIS Controls:

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

1.21 Ensure that IAM Access analyzer is enabled (Automated)

Profile Applicability:

- Level 1

Description:

Enable IAM Access analyzer for IAM policies about all resources.

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.

Rationale:

AWS IAM Access Analyzer helps you identify the resources in your organization and accounts, such as Amazon S3 buckets or IAM roles, that are shared with an external entity. This lets you identify unintended access to your resources and data. Access Analyzer identifies resources that are shared with external principals by using logic-based reasoning to analyze the resource-based policies in your AWS environment. IAM Access Analyzer continuously monitors all policies for S3 bucket, IAM roles, KMS(Key Management Service) keys, AWS Lambda functions, and Amazon SQS(Simple Queue Service) queues.

Audit:

From Console:

1. Open the IAM console at https://console.aws.amazon.com/iam/
2. Choose Access analyzer
3. Ensure that the STATUS is set to Active

From Command Line:

1. Run the following command:

```
aws accessanalyzer get-analyzer --analyzer-name | grep status
```

2. Ensure that the "status" is set to "ACTIVE"

Remediation:

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.
6. Optional. Add any tags that you want to apply to the analyzer.
7. Choose Create Analyzer.

From Command Line:
Run the following command:

```
aws accessanalyzer create-analyzer --analyzer-name --type
```

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

References:

1. https://docs.aws.amazon.com/IAM/latest/UserGuide/what-is-access-analyzer.html
2. https://docs.aws.amazon.com/IAM/latest/UserGuide/access-analyzer-getting-started.html
3. https://docs.aws.amazon.com/cli/latest/reference/accessanalyzer/get-analyzer.html
4. https://docs.aws.amazon.com/cli/latest/reference/accessanalyzer/create-analyzer.html

CIS Controls:

Version 7

14 Controlled Access Based on the Need to Know
Controlled Access Based on the Need to Know

14.6 Protect Information through Access Control Lists
Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

1.22 Ensure IAM users are managed centrally via identity federation or AWS Organizations for multi-account environments (Manual)

Profile Applicability:

- Level 2

Description:

In multi-account environments, IAM user centralization facilitates greater user control. User access beyond the initial account is then provide via role assumption. Centralization of users can be accomplished through federation with an external identity provider or through the use of AWS Organizations.

Rationale:

Centralizing IAM user management to a single identity store reduces complexity and thus the likelihood of access management errors.

Audit:

For multi-account AWS environments with an external identity provider...

1. Determine the master account for identity federation or IAM user management
2. Login to that account through the AWS Management Console
3. Click `Services`
4. Click `IAM`
5. Click `Identity providers`
6. Verify the configuration

Then..., determine all accounts that should not have local users present. For each account...

1. Determine all accounts that should not have local users present
2. Log into the AWS Management Console
3. Switch role into each identified account
4. Click `Services`
5. Click `IAM`
6. Click `Users`
7. Confirm that no IAM users representing individuals are present

For multi-account AWS environments implementing AWS Organizations without an external identity provider...

1. Determine all accounts that should not have local users present

2. Log into the AWS Management Console
3. Switch role into each identified account
4. Click `Services`
5. Click `IAM`
6. Click `Users`
7. Confirm that no IAM users representing individuals are present

Remediation:

The remediation procedure will vary based on the individual organization's implementation of identity federation and/or AWS Organizations with the acceptance criteria that no non-service IAM users, and non-root accounts, are present outside the account providing centralized IAM user management.

CIS Controls:

Version 7

16.2 Configure Centralized Point of Authentication

Configure access for all accounts through as few centralized points of authentication as possible, including network, security, and cloud systems.

2 Storage

2.1 Simple Storage Service (S3)

2.1.1 Ensure all S3 buckets employ encryption-at-rest (Manual)

Profile Applicability:

- Level 1
- Level 2

Description:

Amazon S3 provides a variety of no, or low, cost encryption options to protect data at rest.

Rationale:

Encrypting data at rest reduces the likelihood that it is unintentionally exposed and can nullify the impact of disclosure if the encryption remains unbroken.

Impact:

Amazon S3 buckets with default bucket encryption using SSE-KMS cannot be used as destination buckets for Amazon S3 server access logging. Only SSE-S3 default encryption is supported for server access log destination buckets.

Audit:

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 'Properties'.
4. Verify that `Default Encryption` displays either `AES-256` or `AWS-KMS`.
5. Repeat for all the buckets in your AWS account.

From Command Line:

1. Run

```
aws s3 ls
```

2. For each bucket, run

```
aws s3api get-bucket-encryption --bucket <bucket name>
```

3. Verify that either

```
"SSEAlgorithm": "AES256"
```

Or

```
 "SSEAlgorithm": "aws:kms"
```

is displayed.

Remediation:

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 'Properties'.
4. Click on `Default Encryption`.
5. Select either `AES-256` or `AWS-KMS`
6. Click `Save`
7. Repeat for all the buckets in your AWS account lacking encryption.

From Command Line:
Run either

```
aws s3api put-bucket-encryption --bucket <bucket name> --server-side-
encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault":
{"SSEAlgorithm": "AES256"}}]}'
```

or

```
aws s3api put-bucket-encryption --bucket <bucket name> --server-side-
encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault":
{"SSEAlgorithm": "aws:kms","KMSMasterKeyID": "aws/s3"}}]}'
```

Note: the KMSMasterKeyID can be set to the master key of your choosing; aws/s3 is an AWS preconfigured default.

References:

1. https://docs.aws.amazon.com/AmazonS3/latest/user-guide/default-bucket-encryption.html
2. https://docs.aws.amazon.com/AmazonS3/latest/dev/bucket-encryption.html#bucket-encryption-related-resources

Additional Information:

S3 bucket encryption only applies to objects as they are placed in the bucket. Enabling S3 bucket encryption does **not** encrypt objects previously stored within the bucket.

CIS Controls:

Version 7

14.8 Encrypt Sensitive Information at Rest

Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.

2.1.2 Ensure S3 Bucket Policy allows HTTPS requests (Manual)

Profile Applicability:

- Level 2

Description:

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

Rationale:

By default, Amazon S3 allows both HTTP and HTTPS requests. To achieve only allowing access to Amazon S3 objects through HTTPS you also have to explicitly deny access to HTTP requests. Bucket policies that allow HTTPS requests without explicitly denying HTTP requests will not comply with this recommendation.

Audit:

To allow access to HTTPS you can use a condition that checks for the key `"aws:SecureTransport: true"`. This means that the request is sent through HTTPS but that HTTP can still be used. So to make sure you do not allow HTTP access confirm that there is a bucket policy that explicitly denies access for HTTP requests and that it contains the key "aws:SecureTransport": "false".

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', then Click on `Bucket Policy`.
4. Ensure that a policy is listed that matches:

```
'{
"Sid": ,
"Effect": "Deny",
"Principal": "",
"Action": "s3:GetObject",
"Resource": "arn:aws:s3:::<bucket_name>/",
"Condition": {
"Bool": {
"aws:SecureTransport": "false"
}'
```

   `<optional>` and `<bucket_name>` will be specific to your account'

5. Repeat for all the buckets in your AWS account.

From Command Line:

1. List all of the S3 Buckets

```
aws s3 ls
```

2. Using the list of buckets run this command on each of them:

```
aws s3api get-bucket-policy --bucket <bucket_name> | grep aws:SecureTransport
```

3. Confirm that `aws:SecureTransport` is set to false `aws:SecureTransport:false`
4. Confirm that the policy line has Effect set to Deny 'Effect:Deny'

Remediation:

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:GetObject",
            "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` = GetObject
   `Amazon Resource Name` =
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:GetObject",
            "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
```

References:

1. https://aws.amazon.com/premiumsupport/knowledge-center/s3-bucket-policy-for-config-rule/
2. https://aws.amazon.com/blogs/security/how-to-use-bucket-policies-and-apply-defense-in-depth-to-help-secure-your-amazon-s3-data/
3. https://awscli.amazonaws.com/v2/documentation/api/latest/reference/s3api/get-bucket-policy.html

CIS Controls:

Version 7

14.4 Encrypt All Sensitive Information in Transit
Encrypt all sensitive information in transit.

2.2 Elastic Compute Cloud (EC2)

2.2.1 Ensure EBS volume encryption is enabled (Manual)

Profile Applicability:

- Level 1
- Level 2

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.

Rationale:

Encrypting data at rest reduces the likelihood that it is unintentionally exposed and can nullify the impact of disclosure if the encryption remains unbroken.

Audit:

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. Verify `Always encrypt new EBS volumes` displays `Enabled`.
4. Review every region in-use.

Note: EBS volume encryption is configured per region.
From Command Line:

1. Run

```
aws --region <region> ec2 get-ebs-encryption-by-default.
```

2. Verify that `"EbsEncryptionByDefault": true` is displayed.
3. Review every region in-use.

Note: EBS volume encryption is configured per region.

Remediation:

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.

References:

1. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html
2. https://aws.amazon.com/blogs/aws/new-opt-in-to-default-encryption-for-new-ebs-volumes/

Additional Information:

Default EBS volume encryption only applies to newly created EBS volumes. Existing EBS volumes are **not** converted automatically.

CIS Controls:

Version 7

14.8 Encrypt Sensitive Information at Rest
Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.

3 Logging

This section contains recommendations for configuring AWS's account logging features.

3.1 Ensure CloudTrail is enabled in all regions (Automated)

Profile Applicability:

- Level 1

Description:

AWS CloudTrail is a web service that records AWS API calls for your account and delivers log files to you. 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 provides a history of AWS API calls for an account, including API calls made via the Management Console, SDKs, command line tools, and higher-level AWS services (such as CloudFormation).

Rationale:

The AWS API call history produced by CloudTrail enables security analysis, resource change tracking, and compliance auditing. Additionally,

- ensuring that a multi-regions trail exists will ensure that unexpected activity occurring in otherwise unused regions is detected
- ensuring that a multi-regions trail exists will ensure that `Global Service Logging` is enabled for a trail by default to capture recording of events generated on AWS global services
- for a multi-regions trail, ensuring that management events configured for all type of Read/Writes ensures recording of management operations that are performed on all resources in an AWS account

Impact:

S3 lifecycle features can be used to manage the accumulation and management of logs over time. See the following AWS resource for more information on these features:

1. https://docs.aws.amazon.com/AmazonS3/latest/dev/object-lifecycle-mgmt.html

Audit:

Perform the following to determine if CloudTrail is enabled for all regions:
From Console:

1. Sign in to the AWS Management Console and open the CloudTrail console at https://console.aws.amazon.com/cloudtrail
2. Click on `Trails` on the left navigation pane

- You will be presented with a list of trails across all regions

3. Ensure at least one Trail has `All` specified in the `Region` column
4. Click on a trail via the link in the *Name* column
5. Ensure `Logging` is set to `ON`
6. Ensure `Apply trail to all regions` is set to `Yes`
7. In section `Management Events` ensure `Read/Write Events` set to `ALL`

From Command Line:

```
aws cloudtrail describe-trails
```

Ensure `IsMultiRegionTrail` is set to `true`

```
aws cloudtrail get-trail-status --name <trailname shown in describe-trails>
```

Ensure `IsLogging` is set to `true`

```
aws cloudtrail get-event-selectors --trail-name <trailname shown in describe-trails>
```

Ensure there is at least one Event Selector for a Trail with `IncludeManagementEvents` set to `true` and `ReadWriteType` set to `All`

Remediation:

Perform the following to enable global (Multi-region) CloudTrail logging:
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 `Get Started Now` , if presented

   - Click `Add new trail`
   - Enter a trail name in the `Trail name` box
   - Set the `Apply trail to all regions` option to `Yes`
   - Specify an S3 bucket name in the `S3 bucket` box
   - Click `Create`

4. If 1 or more trails already exist, select the target trail to enable for global logging
5. Click the edit icon (pencil) next to `Apply trail to all regions` , Click `Yes` and Click `Save`.
6. Click the edit icon (pencil) next to `Management Events` click `All` for setting `Read/Write Events` and Click `Save`.

From Command Line:

```
aws cloudtrail create-trail --name <trail_name> --bucket-name
<s3_bucket_for_cloudtrail> --is-multi-region-trail
aws cloudtrail update-trail --name <trail_name> --is-multi-region-trail
```

Note: Creating CloudTrail via CLI without providing any overriding options configures `Management Events` to set `All` type of `Read/Writes` by default.

Default Value:

Not Enabled

References:

1. CCE-78913-1
2. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-concepts.html#cloudtrail-concepts-management-events
3. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/logging-management-and-data-events-with-cloudtrail.html?icmpid=docs_cloudtrail_console#logging-management-events
4. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-supported-services.html#cloud-trail-supported-services-data-events

CIS Controls:

Version 7

6.2 Activate audit logging
Ensure that local logging has been enabled on all systems and networking devices.

3.2 Ensure CloudTrail log file validation is enabled (Automated)

Profile Applicability:

- Level 2

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.

Rationale:

Enabling log file validation will provide additional integrity checking of CloudTrail logs.

Audit:

Perform the following on each trail to determine if log file validation is enabled:
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. For Every Trail:

    - Click on a trail via the link in the *Name* column
    - Under the `S3` section, ensure `Enable log file validation` is set to `Yes`

From Command Line:

```
aws cloudtrail describe-trails
```

Ensure `LogFileValidationEnabled` is set to `true` for each trail

Remediation:

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 `S3` section click on the edit icon (pencil)
5. Click `Advanced`
6. Click on the `Yes` radio button in section `Enable log file validation`
7. Click `Save`

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>
```

Default Value:

Not Enabled

References:

1. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-log-file-validation-enabling.html
2. CCE-78914-9

CIS Controls:

Version 7

6 Maintenance, Monitoring and Analysis of Audit Logs
Maintenance, Monitoring and Analysis of Audit Logs

3.3 Ensure the S3 bucket used to store CloudTrail logs is not publicly accessible (Automated)

Profile Applicability:

- Level 1

Description:

CloudTrail logs a record of every API call made in your AWS account. These logs file are stored in an S3 bucket. It is recommended that the bucket policy or access control list (ACL) applied to the S3 bucket that CloudTrail logs to prevent public access to the CloudTrail logs.

Rationale:

Allowing public access to CloudTrail log content may aid an adversary in identifying weaknesses in the affected account's use or configuration.

Audit:

Perform the following to determine if any public access is granted to an S3 bucket via an ACL or S3 bucket policy:

From Console:

1. Go to the Amazon CloudTrail console at https://console.aws.amazon.com/cloudtrail/home
2. In the `API activity history` pane on the left, click `Trails`
3. In the `Trails` pane, note the bucket names in the `S3 bucket` column
4. Go to Amazon S3 console at https://console.aws.amazon.com/s3/home
5. For each bucket noted in step 3, right-click on the bucket and click `Properties`
6. In the `Properties` pane, click the `Permissions` tab.
7. The tab shows a list of grants, one row per grant, in the bucket ACL. Each row identifies the grantee and the permissions granted.
8. Ensure no rows exists that have the `Grantee` set to `Everyone` or the `Grantee` set to `Any Authenticated User.`
9. If the `Edit bucket policy` button is present, click it to review the bucket policy.
10. Ensure the policy does not contain a `Statement` having an `Effect` set to `Allow` and a `Principal` set to "*" or {"AWS" : "*"}

From Command Line:

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

```
aws cloudtrail describe-trails --query 'trailList[*].S3BucketName'
```

2. Ensure the `AllUsers` principal is not granted privileges to that `<bucket>` :

```
  aws s3api get-bucket-acl --bucket <s3_bucket_for_cloudtrail> --query
'Grants[?Grantee.URI== `https://acs.amazonaws.com/groups/global/AllUsers` ]'
```

3. Ensure the `AuthenticatedUsers` principal is not granted privileges to that `<bucket>`:

```
  aws s3api get-bucket-acl --bucket <s3_bucket_for_cloudtrail> --query
'Grants[?Grantee.URI== `https://acs.amazonaws.com/groups/global/Authenticated
Users` ]'
```

4. Get the S3 Bucket Policy

```
  aws s3api get-bucket-policy --bucket <s3_bucket_for_cloudtrail>
```

5. Ensure the policy does not contain a `Statement` having an `Effect` set to `Allow` and a `Principal` set to "*" or {"AWS" : "*"}

Note: Principal set to "*" or {"AWS" : "*"} allows anonymous access.

Remediation:

Perform the following to remove any public access that has been granted to the bucket via an ACL or S3 bucket policy:

1. Go to Amazon S3 console at https://console.aws.amazon.com/s3/home
2. Right-click on the bucket and click Properties
3. In the `Properties` pane, click the `Permissions` tab.
4. The tab shows a list of grants, one row per grant, in the bucket ACL. Each row identifies the grantee and the permissions granted.
5. Select the row that grants permission to `Everyone` or `Any Authenticated User`
6. Uncheck all the permissions granted to `Everyone` or `Any Authenticated User` (click `x` to delete the row).
7. Click `Save` to save the ACL.
8. If the `Edit bucket policy` button is present, click it.
9. Remove any `Statement` having an `Effect` set to `Allow` and a `Principal` set to "*" or {"AWS" : "*"}.

Default Value:

By default, S3 buckets are not publicly accessible

References:

1. CCE-78915-6

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

CIS Controls:

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

3.4 Ensure CloudTrail trails are integrated with CloudWatch Logs (Automated)

Profile Applicability:

- Level 1

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, realtime 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.

Rationale:

Sending CloudTrail logs to CloudWatch Logs will facilitate real-time and historic activity logging based on user, API, resource, and IP address, and provides opportunity to establish alarms and notifications for anomalous or sensitivity account activity.

Impact:

Note: By default, CloudWatch Logs will store Logs indefinitely unless a specific retention period is defined for the log group. When choosing the number of days to retain, keep in mind the average days it takes an organization to realize they have been breached is 210 days (at the time of this writing). Since additional time is required to research a breach, a minimum 365 day retention policy allows time for detection and research. You may also wish to archive the logs to a cheaper storage service rather than simply deleting them. See the following AWS resource to manage CloudWatch Logs retention periods:

1. https://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/SettingLogRetention.html

Audit:

Perform the following to ensure CloudTrail is configured as prescribed:
From Console:

1. Sign in to the AWS Management Console and open the CloudTrail console at https://console.aws.amazon.com/cloudtrail/
2. Under `All Buckets` , click on the target bucket you wish to evaluate
3. Click `Properties` on the top right of the console
4. Click `Trails` in the left menu
5. Ensure a `CloudWatch Logs` log group is configured and has a recent (~one day old) `Last log file delivered` timestamp.

From Command Line:

1. Run the following command to get a listing of existing trails:

```
aws cloudtrail describe-trails
```

2. Ensure `CloudWatchLogsLogGroupArn` is not empty and note the value of the `Name` property.
3. Using the noted value of the `Name` property, run the following command:

```
aws cloudtrail get-trail-status --name <trail_name>
```

4. Ensure the `LatestcloudwatchLogdDeliveryTime` property is set to a recent (~one day old) timestamp.

Remediation:

Perform the following to establish the prescribed state:
From Console:

1. Sign in to the AWS Management Console and open the CloudTrail console at https://console.aws.amazon.com/cloudtrail/
2. Under All Buckets, click on the target bucket you wish to evaluate
3. Click Properties on the top right of the console
4. Click `Trails` in the left menu
5. Click on each trail where no `CloudWatch Logs` are defined
6. Go to the `CloudWatch Logs` section and click on `Configure`
7. Define a new or select an existing log group
8. Click on `Continue`
9. Configure IAM Role which will deliver CloudTrail events to CloudWatch Logs
    - Create/Select an `IAM Role` and `Policy Name`
    - Click `Allow` to continue

From Command Line:

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

References:

1. https://aws.amazon.com/cloudtrail/
2. CCE-78916-4

CIS Controls:

Version 7

6.2 Activate audit logging

Ensure that local logging has been enabled on all systems and networking devices.

6.5 Central Log Management

Ensure that appropriate logs are being aggregated to a central log management system for analysis and review.

3.5 Ensure AWS Config is enabled in all regions (Automated)

Profile Applicability:

- Level 1

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 to enable AWS Config be enabled in all regions.

Rationale:

The AWS configuration item history captured by AWS Config enables security analysis, resource change tracking, and compliance auditing.

Audit:

Process to evaluate AWS Config configuration per region
From Console:

1. Sign in to the AWS Management Console and open the AWS Config console at https://console.aws.amazon.com/config/.
2. On the top right of the console select target Region.
3. If presented with Setup AWS Config - follow remediation procedure:
4. On the Resource inventory page, Click on edit (the gear icon). The Set Up AWS Config page appears.
5. Ensure 1 or both check-boxes under "All Resources" is checked.
    - Include global resources related to IAM resources - which needs to be enabled in 1 region only
6. Ensure the correct S3 bucket has been defined.
7. Ensure the correct SNS topic has been defined.
8. Repeat steps 2 to 7 for each region.

From Command Line:

1. Run this command to show all AWS Config recorders and their properties:

```
aws configservice describe-configuration-recorders
```

2. Evaluate the output to ensure that there's at least one recorder for which `recordingGroup` object includes `"allSupported": true` AND `"includeGlobalResourceTypes": true`

Note: There is one more parameter "ResourceTypes" in recordingGroup object. We don't need to check the same as whenever we set "allSupported": true, AWS enforces resource types to be empty ("ResourceTypes":[])
Sample Output:

```
{
    "ConfigurationRecorders": [
        {
            "recordingGroup": {
                "allSupported": true,
                "resourceTypes": [],
                "includeGlobalResourceTypes": true
            },
            "roleARN": "arn:aws:iam::<AWS_Account_ID>:role/service-
role/<config-role-name>",
            "name": "default"
        }
    ]
}
```

3. Run this command to show the status for all AWS Config recorders:

```
aws configservice describe-configuration-recorder-status
```

4. In the output, find recorders with `name` key matching the recorders that met criteria in step 2. Ensure that at least one of them includes `"recording": true` and `"lastStatus": "SUCCESS"`

Remediation:

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. Define which resources you want to record in the selected region
5. Choose to include global resources (IAM resources)
6. Specify an S3 bucket in the same account or in another managed AWS account
7. 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 set up the configuration recorder

```
aws configservice subscribe --s3-bucket my-config-bucket --sns-topic
arn:aws:sns:us-east-1:012345678912:my-config-notice --iam-role
arn:aws:iam::012345678912:role/myConfigRole
```

3. Run this command to start the configuration recorder:

```
start-configuration-recorder --configuration-recorder-name <value>
```

References:

1. CCE-78917-2
2. https://docs.aws.amazon.com/cli/latest/reference/configservice/describe-configuration-recorder-status.html

CIS Controls:

Version 7

1.4 Maintain Detailed Asset Inventory

Maintain an accurate and up-to-date inventory of all technology assets with the potential to store or process information. This inventory shall include all hardware assets, whether connected to the organization's network or not.

11.2 Document Traffic Configuration Rules

All configuration rules that allow traffic to flow through network devices should be documented in a configuration management system with a specific business reason for each rule, a specific individual's name responsible for that business need, and an expected duration of the need.

16.1 Maintain an Inventory of Authentication Systems

Maintain an inventory of each of the organization's authentication systems, including those located onsite or at a remote service provider.

3.6 Ensure S3 bucket access logging is enabled on the CloudTrail S3 bucket (Automated)

Profile Applicability:

- Level 1

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.

Rationale:

By enabling S3 bucket logging on target S3 buckets, it is possible to capture all events which may affect objects within any target buckets. Configuring logs to be placed in a separate bucket allows access to log information which can be useful in security and incident response workflows.

Audit:

Perform the following ensure the CloudTrail S3 bucket has access logging is enabled:
From Console:

1. Go to the Amazon CloudTrail console at https://console.aws.amazon.com/cloudtrail/home
2. In the API activity history pane on the left, click Trails
3. In the Trails pane, note the bucket names in the S3 bucket column
4. Sign in to the AWS Management Console and open the S3 console at https://console.aws.amazon.com/s3.
5. Under `All Buckets` click on a target S3 bucket
6. Click on `Properties` in the top right of the console
7. Under `Bucket: _ <bucket_name> _` click on `Logging`
8. Ensure `Enabled` is checked.

From Command Line:

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

```
aws cloudtrail describe-trails --query 'trailList[*].S3BucketName'
```

2. Ensure Bucket Logging is enabled:

```
aws s3api get-bucket-logging --bucket <s3_bucket_for_cloudtrail>
```

Ensure command does not returns empty output.
Sample Output for a bucket with logging enabled:

```
{
    "LoggingEnabled": {
        "TargetPrefix": "<Prefix_Test>",
        "TargetBucket": "<Bucket_name_for_Storing_Logs>"
    }
}
```

Remediation:

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 `Enabled` checkbox
    - Select Target Bucket from list
    - Enter a Target Prefix

6. Click `Save`

Default Value:

Logging is disabled.

References:

1. CCE-78918-0
2. https://docs.aws.amazon.com/AmazonS3/latest/dev/ServerLogs.html

CIS Controls:

Version 7

6.2 Activate audit logging
Ensure that local logging has been enabled on all systems and networking devices.

14.9 Enforce Detail Logging for Access or Changes to Sensitive Data

Enforce detailed audit logging for access to sensitive data or changes to sensitive data (utilizing tools such as File Integrity Monitoring or Security Information and Event Monitoring).

3.7 Ensure CloudTrail logs are encrypted at rest using KMS CMKs (Automated)

Profile Applicability:

- Level 2

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.

Rationale:

Configuring CloudTrail to use SSE-KMS provides additional confidentiality controls on log data as a given user must have S3 read permission on the corresponding log bucket and must be granted decrypt permission by the CMK policy.

Impact:

Customer created keys incur an additional cost. See https://aws.amazon.com/kms/pricing/ for more information.

Audit:

Perform the following to determine if CloudTrail is configured 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. Select a Trail
4. Under the `S3` section, ensure `Encrypt log files` is set to `Yes` and a KMS key ID is specified in the `KSM Key Id` field.

From Command Line:

1. Run the following command:

```
aws cloudtrail describe-trails
```

2. For each trail listed, SSE-KMS is enabled if the trail has a `KmsKeyId` property defined.

Remediation:

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>
```

References:

1. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/encrypting-cloudtrail-log-files-with-aws-kms.html
2. https://docs.aws.amazon.com/kms/latest/developerguide/create-keys.html
3. CCE-78919-8

Additional Information:

3 statements which need to be added to the CMK policy:

1. Enable Cloudtrail to describe CMK properties

```
<pre class="programlisting" style="font-style: normal;">{
  "Sid": "Allow CloudTrail access",
  "Effect": "Allow",
  "Principal": {
    "Service": "cloudtrail.amazonaws.com"
  },
  "Action": "kms:DescribeKey",
  "Resource": "*"
}
```

2. Granting encrypt permissions

```
<pre class="programlisting" style="font-style: normal;">{
  "Sid": "Allow CloudTrail to encrypt logs",
  "Effect": "Allow",
  "Principal": {
    "Service": "cloudtrail.amazonaws.com"
  },
  "Action": "kms:GenerateDataKey*",
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "kms:EncryptionContext:aws:cloudtrail:arn": [
        "arn:aws:cloudtrail:*:aws-account-id:trail/*"
      ]
    }
```

```
  }
}
```

3. Granting decrypt permissions

```
<pre class="programlisting" style="font-style: normal;">{
  "Sid": "Enable CloudTrail log decrypt permissions",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::aws-account-id:user/username"
  },
  "Action": "kms:Decrypt",
  "Resource": "*",
  "Condition": {
    "Null": {
      "kms:EncryptionContext:aws:cloudtrail:arn": "false"
    }
  }
}
```

CIS Controls:

Version 7

6 Maintenance, Monitoring and Analysis of Audit Logs
Maintenance, Monitoring and Analysis of Audit Logs

3.8 Ensure rotation for customer created CMKs is enabled (Automated)

Profile Applicability:

- Level 2

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.

Rationale:

Rotating encryption keys helps reduce the potential impact of a compromised key as data encrypted with a new key cannot be accessed with a previous key that may have been exposed.

Audit:

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 `Encryption Keys`.
3. Select a customer created master key (CMK)
4. Under the `Key Policy` section, move down to `Key Rotation`.
5. Ensure the `Rotate this key every year` checkbox is checked.

From Command Line:

1. Run the following command to get a list of all keys and their associated `KeyIds`

```
aws kms list-keys
```

2. For each key, note the KeyId and run the following command

```
aws kms get-key-rotation-status --key-id <kms_key_id>
```

3. Ensure `KeyRotationEnabled` is set to `true`

Remediation:

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 `Encryption Keys`.
3. Select a customer created master key (CMK)
4. Under the `Key Policy` section, move down to `Key Rotation`.
5. Check the `Rotate this 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>
```

References:

1. https://aws.amazon.com/kms/pricing/
2. https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-5/final
3. CCE-78920-6

CIS Controls:

Version 7

6 Maintenance, Monitoring and Analysis of Audit Logs
Maintenance, Monitoring and Analysis of Audit Logs

3.9 Ensure VPC flow logging is enabled in all VPCs (Automated)

Profile Applicability:

- Level 2

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.

Rationale:

VPC Flow Logs provide visibility into network traffic that traverses the VPC and can be used to detect anomalous traffic or insight during security workflows.

Impact:

By default, CloudWatch Logs will store Logs indefinitely unless a specific retention period is defined for the log group. When choosing the number of days to retain, keep in mind the average days it takes an organization to realize they have been breached is 210 days (at the time of this writing). Since additional time is required to research a breach, a minimum 365 day retention policy allows time for detection and research. You may also wish to archive the logs to a cheaper storage service rather than simply deleting them. See the following AWS resource to manage CloudWatch Logs retention periods:

1. https://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/SettingLogRetention.html

Audit:

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. Ensure a Log Flow exists that has `Active` in the `Status` column.

Remediation:

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.

References:

1. CCE-79202-8
2. https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/flow-logs.html

CIS Controls:

Version 7

6.2 Activate audit logging
Ensure that local logging has been enabled on all systems and networking devices.

12.5 Configure Monitoring Systems to Record Network Packets
Configure monitoring systems to record network packets passing through the boundary at each of the organization's network boundaries.

3.10 Ensure that Object-level logging for write events is enabled for S3 bucket (Automated)

Profile Applicability:

- Level 2

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.

Rationale:

Enabling object-level logging will help you meet data compliance requirements within your organization, perform comprehensive security analysis, monitor specific patterns of user behavior in your AWS account or take immediate actions on any object-level API activity within your S3 Buckets using Amazon CloudWatch Events.

Audit:

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. If the current status for `Object-level` logging is set to Disabled, then object-level logging of write events for the selected s3 bucket is not set.
5. Repeat steps 2 to 4 to verify object level logging status of other S3 buckets.

From Command Line:

1. Run `list-trails` command to list the names of all Amazon CloudTrail trails currently available in the selected AWS region:

```
aws cloudtrail list-trails --region <region-name> --query Trails[*].Name
```

2. The command output will be a list of the requested trail names.
3. Run `get-event-selectors` command using the name of the trail returned at the previous step and custom query filters to determine if Data events logging feature is enabled within the selected CloudTrail trail configuration for s3bucket resources:

```
aws cloudtrail get-event-selectors --region <region-name> --trail-name
<trail-name> --query EventSelectors[*].DataResources[]
```

4. The command output should be an array that contains the configuration of the AWS resource(S3 bucket) defined for the Data events selector.
5. If the `get-event-selectors` command returns an empty array '[]', the Data events are not included into the selected AWS Cloudtrail trail logging configuration, therefore the S3 object-level API operations performed within your AWS account are not recorded.
6. Repeat steps 1 to 5 for auditing each s3 bucket to identify other trails that are missing the capability to log Data events.
7. Change the AWS region by updating the `--region` command parameter and perform the audit process for other regions.

Remediation:

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 `object-level` 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.

References:

1. https://docs.aws.amazon.com/AmazonS3/latest/user-guide/enable-cloudtrail-events.html

CIS Controls:

Version 7

6.2 Activate audit logging
Ensure that local logging has been enabled on all systems and networking devices.

6.3 Enable Detailed Logging
Enable system logging to include detailed information such as an event source, date, user, timestamp, source addresses, destination addresses, and other useful elements.

3.11 Ensure that Object-level logging for read events is enabled for S3 bucket (Automated)

Profile Applicability:

- Level 2

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.

Rationale:

Enabling object-level logging will help you meet data compliance requirements within your organization, perform comprehensive security analysis, monitor specific patterns of user behavior in your AWS account or take immediate actions on any object-level API activity using Amazon CloudWatch Events.

Audit:

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. If the current status for `Object-level` logging is set to `Disabled`, then object-level logging of read events for the selected s3 bucket is not set.
5. If the current status for `Object-level` logging is set to `Enabled`, but the Read event check-box is unchecked, then object-level logging of read events for the selected s3 bucket is not set.
6. Repeat steps 2 to 5 to verify `object-level` logging for `read` events of your other S3 buckets.

From Command Line:

1. Run `describe-trails` command to list the names of all Amazon CloudTrail trails currently available in the selected AWS region:

```
aws cloudtrail describe-trails --region <region-name> --output table --query
trailList[*].Name
```

2. The command output will be table of the requested trail names.
3. Run `get-event-selectors` command using the name of the trail returned at the previous step and custom query filters to determine if Data events logging feature is enabled within the selected CloudTrail trail configuration for s3 bucket resources:

```
aws cloudtrail get-event-selectors --region <region-name> --trail-name
<trail-name> --query EventSelectors[*].DataResources[]
```

4. The command output should be an array that contains the configuration of the AWS resource(S3 bucket) defined for the Data events selector.
5. If the `get-event-selectors` command returns an empty array, the Data events are not included into the selected AWS Cloudtrail trail logging configuration, therefore the S3 object-level API operations performed within your AWS account are not recorded.
6. Repeat steps 1 to 5 for auditing each s3 bucket to identify other trails that are missing the capability to log Data events.
7. Change the AWS region by updating the `--region` command parameter and perform the audit process for other regions.

Remediation:

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 `object-level` 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 ones 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.

References:

1. https://docs.aws.amazon.com/AmazonS3/latest/user-guide/enable-cloudtrail-events.html

CIS Controls:

Version 7

6.2 Activate audit logging

Ensure that local logging has been enabled on all systems and networking devices.

6.3 Enable Detailed Logging

Enable system logging to include detailed information such as an event source, date, user, timestamp, source addresses, destination addresses, and other useful elements.

4 Monitoring

For effectiveness and coverage of recommended metric-filters and alarms, recommendations in Section 3 should be implemented on Multi-region CloudTrail referred in `Ensure CloudTrail is enabled in all regions` Updated Overview should look like: This section contains recommendations for configuring AWS to assist with monitoring and responding to account activities. Metric filter-related recommendations in this section are dependent on the `Ensure CloudTrail is enabled in all regions` and `Ensure CloudTrail trails are integrated with CloudWatch Logs` recommendation in the "Logging" section. Additionally, step 3 of the remediation procedure for the same recommendations provides guidance for establishing an email-based subscription (`--protocol email`). This is provided as an example and is not meant to suggest other protocols provide lesser value.

4.1 Ensure a log metric filter and alarm exist for unauthorized API calls (Automated)

Profile Applicability:

- Level 1

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 unauthorized API calls.

Rationale:

Monitoring unauthorized API calls will help reveal application errors and may reduce time to detect malicious activity.

Impact:

This alert may be triggered by normal read-only console activities that attempt to opportunistically gather optional information, but gracefully fail if they don't have permissions.

If an excessive number of alerts are being generated then an organization may wish to consider adding read access to the limited IAM user permissions simply to quiet the alerts.

In some cases doing this may allow the users to actually view some areas of the system - any additional access given should be reviewed for alignment with the original limited IAM user intent.

Audit:

Perform the following to ensure that there is at least one active multi-region CloudTrail with prescribed metric filters and alarms configured:

1. Identify the log group name configured for use with active multi-region CloudTrail:

- List all CloudTrails: `aws cloudtrail describe-trails`
- Identify Multi region Cloudtrails: `Trails with "IsMultiRegionTrail" set to true`
- From value associated with CloudWatchLogsLogGroupArn note `<cloudtrail_log_group_name>`

Example: for CloudWatchLogsLogGroupArn that looks like `arn:aws:logs:<region>:<aws_account_number>:log-group:NewGroup:*`, `<cloudtrail_log_group_name>` would be `NewGroup`

- Ensure Identified Multi region CloudTrail is active

`aws cloudtrail get-trail-status --name <Name of a Multi-region CloudTrail>` ensure `IsLogging` is set to `TRUE`

- Ensure identified Multi-region Cloudtrail captures all Management Events

`aws cloudtrail get-event-selectors --trail-name <trailname shown in describe-trails>`
Ensure there is at least one Event Selector for a Trail with `IncludeManagementEvents` set to `true` and `ReadWriteType` set to `All`

2. Get a list of all associated metric filters for this `<cloudtrail_log_group_name>`:

```
aws logs describe-metric-filters --log-group-name
"<cloudtrail_log_group_name>"
```

3. Ensure the output from the above command contains the following:

```
"Filter = {(($.errorCode="UnauthorizedOperation") ||
($.errorCode="AccessDenied")) ||
(($.sourceIPAddress!="delivery.logs.amazonaws.com") ||
($.eventName!="HeadBucket"))}"
```

4. Note the `<unauthorized_api_calls_metric>` value associated with the `filterPattern` found in step 3.
5. Get a list of CloudWatch alarms and filter on the `<unauthorized_api_calls_metric>` captured in step 4.

```
aws cloudwatch describe-alarms --query 'MetricAlarms[?MetricName==
`<unauthorized_api_calls_metric>`]'
```

6. Note the `AlarmActions` value - this will provide the SNS topic ARN value.
7. Ensure there is at least one active subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <sns_topic_arn>
```

at least one subscription should have "SubscriptionArn" with valid aws ARN.

```
Example of valid "SubscriptionArn":
"arn:aws:sns:<region>:<aws_account_number>:<SnsTopicName>:<SubscriptionID>"
```

Remediation:

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-transformations
metricName= `<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.

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
`<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>
```

References:

1. https://aws.amazon.com/sns/
2. CCE-79186-3
3. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/receive-cloudtrail-log-files-from-multiple-regions.html
4. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html
5. https://docs.aws.amazon.com/sns/latest/dg/SubscribeTopic.html

Additional Information:

Configuring log metric filter and alarm on Multi-region (global) CloudTrail

- ensures that activities from all regions (used as well as unused) are monitored
- ensures that activities on all supported global services are monitored
- ensures that all management events across all regions are monitored

CIS Controls:

Version 7

6.5 Central Log Management

Ensure that appropriate logs are being aggregated to a central log management system for analysis and review.

6.7 Regularly Review Logs

On a regular basis, review logs to identify anomalies or abnormal events.

4.2 Ensure a log metric filter and alarm exist for Management Console sign-in without MFA (Automated)

Profile Applicability:

- Level 1

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 console logins that are not protected by multi-factor authentication (MFA).

Rationale:

Monitoring for single-factor console logins will increase visibility into accounts that are not protected by MFA.

Audit:

Perform the following to ensure that there is at least one active multi-region CloudTrail with prescribed metric filters and alarms configured:

1. Identify the log group name configured for use with active multi-region CloudTrail:

- List all `CloudTrails`:

```
aws cloudtrail describe-trails
```

- Identify Multi region Cloudtrails: `Trails with "IsMultiRegionTrail" set to true`
- From value associated with CloudWatchLogsLogGroupArn note `<cloudtrail_log_group_name>`

Example: for CloudWatchLogsLogGroupArn that looks like `arn:aws:logs:<region>:<aws_account_number>:log-group:NewGroup:*`, `<cloudtrail_log_group_name>` would be `NewGroup`

- Ensure Identified Multi region `CloudTrail` is active

```
aws cloudtrail get-trail-status --name <Name of a Multi-region CloudTrail>
```

Ensure in the output that `IsLogging` is set to `TRUE`

- Ensure identified Multi-region 'Cloudtrail' captures all Management Events

```
aws cloudtrail get-event-selectors --trail-name <trailname shown in describe-
trails>
```

Ensure in the output there is at least one Event Selector for a Trail with `IncludeManagementEvents` set to `true` and `ReadWriteType` set to `All`

2. Get a list of all associated metric filters for this `<cloudtrail_log_group_name>`:

```
aws logs describe-metric-filters --log-group-name
"<cloudtrail_log_group_name>"
```

3. Ensure the output from the above command contains the following:

```
"filterPattern": "{ ($.eventName = "ConsoleLogin") &&
($.additionalEventData.MFAUsed != "Yes") }"
```

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

```
"filterPattern": "{ ($.eventName = "ConsoleLogin") &&
($.additionalEventData.MFAUsed != "Yes") && ($.userIdentity.type = "IAMUser")
&& ($.responseElements.ConsoleLogin = "Success") }"
```

4. Note the `<no_mfa_console_signin_metric>` value associated with the `filterPattern` found in step 3.
5. Get a list of CloudWatch alarms and filter on the `<no_mfa_console_signin_metric>` captured in step 4.

```
aws cloudwatch describe-alarms --query 'MetricAlarms[?MetricName==
`<no_mfa_console_signin_metric>`]'
```

6. Note the `AlarmActions` value - this will provide the SNS topic ARN value.
7. Ensure there is at least one active subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <sns_topic_arn>
```

at least one subscription should have "SubscriptionArn" with valid aws ARN.

```
Example of valid "SubscriptionArn":
"arn:aws:sns:<region>:<aws_account_number>:<SnsTopicName>:<SubscriptionID>"
```

Remediation:

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-transformations
metricName= `<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>`  --statistic Sum --period 300
--threshold 1 --comparison-operator GreaterThanOrEqualToThreshold --
evaluation-periods 1 --namespace 'CISBenchmark' --alarm-actions
<sns_topic_arn>
```

References:

1. https://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/viewing_metrics_with_cloudwatch.html
2. CCE-79187-1
3. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/receive-cloudtrail-log-files-from-multiple-regions.html
4. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html
5. https://docs.aws.amazon.com/sns/latest/dg/SubscribeTopic.html

Additional Information:

Configuring log metric filter and alarm on Multi-region (global) CloudTrail

- ensures that activities from all regions (used as well as unused) are monitored
- ensures that activities on all supported global services are monitored
- ensures that all management events across all regions are monitored -Filter pattern set to `{ ($.eventName = "ConsoleLogin") && ($.additionalEventData.MFAUsed != "Yes") && ($.userIdentity.type = "IAMUser") && ($.responseElements.ConsoleLogin = "Success")}` reduces false alarms raised when user logs in via SSO account.

CIS Controls:

Version 7

16 Account Monitoring and Control
Account Monitoring and Control

4.3 Ensure a log metric filter and alarm exist for usage of "root" account (Automated)

Profile Applicability:

- Level 1

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 root login attempts.

Rationale:

Monitoring for root account logins will provide visibility into the use of a fully privileged account and an opportunity to reduce the use of it.

Audit:

Perform the following to ensure that there is at least one active multi-region CloudTrail with prescribed metric filters and alarms configured:

1. Identify the log group name configured for use with active multi-region CloudTrail:

- List all CloudTrails:

```
aws cloudtrail describe-trails
```

- Identify Multi region Cloudtrails: `Trails with "IsMultiRegionTrail" set to true`
- From value associated with CloudWatchLogsLogGroupArn note `<cloudtrail_log_group_name>`

Example: for CloudWatchLogsLogGroupArn that looks like `arn:aws:logs:<region>:<aws_account_number>:log-group:NewGroup:*`, `<cloudtrail_log_group_name>` would be `NewGroup`

- Ensure Identified Multi region CloudTrail is active

`aws cloudtrail get-trail-status --name <Name of a Multi-region CloudTrail>` ensure `IsLogging` is set to `TRUE`

- Ensure identified Multi-region Cloudtrail captures all Management Events

```
aws cloudtrail get-event-selectors --trail-name <trailname shown in describe-trails>
```

Ensure there is at least one Event Selector for a Trail with `IncludeManagementEvents` set to `true` and `ReadWriteType` set to `All`

2. Get a list of all associated metric filters for this `<cloudtrail_log_group_name>`:

```
aws logs describe-metric-filters --log-group-name
"<cloudtrail_log_group_name>"
```

3. Ensure the output from the above command contains the following:

```
"filterPattern": "{ $.userIdentity.type = "Root" && $.userIdentity.invokedBy
NOT EXISTS && $.eventType != "AwsServiceEvent" }"
```

4. Note the `<root_usage_metric>` value associated with the `filterPattern` found in step 3.
5. Get a list of CloudWatch alarms and filter on the `<root_usage_metric>` captured in step 4.

```
aws cloudwatch describe-alarms --query 'MetricAlarms[?MetricName==
`<root_usage_metric>`]'
```

6. Note the `AlarmActions` value - this will provide the SNS topic ARN value.
7. Ensure there is at least one active subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <sns_topic_arn>
```

at least one subscription should have "SubscriptionArn" with valid aws ARN.

```
Example of valid "SubscriptionArn":
"arn:aws:sns:<region>:<aws_account_number>:<SnsTopicName>:<SubscriptionID>"
```

Remediation:

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 --filter-
pattern '{ $.userIdentity.type = "Root" && $.userIdentity.invokedBy NOT
EXISTS && $.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>` --metric-name `<root_usage_metric>` --statistic Sum --period 300 --threshold 1 --comparison-operator GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace 'CISBenchmark' --alarm-actions <sns_topic_arn>
```

References:

1. CCE-79188-9
2. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/receive-cloudtrail-log-files-from-multiple-regions.html
3. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html
4. https://docs.aws.amazon.com/sns/latest/dg/SubscribeTopic.html

Additional Information:

Configuring log metric filter and alarm on Multi-region (global) CloudTrail

- ensures that activities from all regions (used as well as unused) are monitored
- ensures that activities on all supported global services are monitored
- ensures that all management events across all regions are monitored

CIS Controls:

Version 7

4.9 Log and Alert on Unsuccessful Administrative Account Login

Configure systems to issue a log entry and alert on unsuccessful logins to an administrative account.

4.4 Ensure a log metric filter and alarm exist for IAM policy changes (Automated)

Profile Applicability:

- Level 1

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 changes made to Identity and Access Management (IAM) policies.

Rationale:

Monitoring changes to IAM policies will help ensure authentication and authorization controls remain intact.

Audit:

Perform the following to ensure that there is at least one active multi-region CloudTrail with prescribed metric filters and alarms configured:

1. Identify the log group name configured for use with active multi-region CloudTrail:

- List all CloudTrails:

```
aws cloudtrail describe-trails
```

- Identify Multi region Cloudtrails: `Trails with "IsMultiRegionTrail" set to true`
- From value associated with CloudWatchLogsLogGroupArn note `<cloudtrail_log_group_name>`

Example: for CloudWatchLogsLogGroupArn that looks like `arn:aws:logs:<region>:<aws_account_number>:log-group:NewGroup:*`, `<cloudtrail_log_group_name>` would be `NewGroup`

- Ensure Identified Multi region CloudTrail is active

```
aws cloudtrail get-trail-status --name <Name of a Multi-region CloudTrail>
```

ensure `IsLogging` is set to `TRUE`

- Ensure identified Multi-region Cloudtrail captures all Management Events

`aws cloudtrail get-event-selectors --trail-name <trailname shown in describe-trails>`

Ensure there is at least one Event Selector for a Trail with `IncludeManagementEvents` set to `true` and `ReadWriteType` set to `All`

2. Get a list of all associated metric filters for this `<cloudtrail_log_group_name>`:

```
aws logs describe-metric-filters --log-group-name
"<cloudtrail_log_group_name>"
```

3. Ensure the output from the above command contains the following:

```
"filterPattern":
"{($.eventName=DeleteGroupPolicy)||($.eventName=DeleteRolePolicy)||($.eventNa
me=DeleteUserPolicy)||($.eventName=PutGroupPolicy)||($.eventName=PutRolePolic
y)||($.eventName=PutUserPolicy)||($.eventName=CreatePolicy)||($.eventName=Del
etePolicy)||($.eventName=CreatePolicyVersion)||($.eventName=DeletePolicyVersi
on)||($.eventName=AttachRolePolicy)||($.eventName=DetachRolePolicy)||($.event
Name=AttachUserPolicy)||($.eventName=DetachUserPolicy)||($.eventName=AttachGr
oupPolicy)||($.eventName=DetachGroupPolicy)}"
```

4. Note the `<iam_changes_metric>` value associated with the `filterPattern` found in step 3.
5. Get a list of CloudWatch alarms and filter on the `<iam_changes_metric>` captured in step 4.

```
aws cloudwatch describe-alarms --query 'MetricAlarms[?MetricName==
`<iam_changes_metric>`]'
```

6. Note the `AlarmActions` value - this will provide the SNS topic ARN value.
7. Ensure there is at least one active subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <sns_topic_arn>
```

at least one subscription should have "SubscriptionArn" with valid aws ARN.

```
Example of valid "SubscriptionArn":
"arn:aws:sns:<region>:<aws_account_number>:<SnsTopicName>:<SubscriptionID>"
```

Remediation:

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)||($.eventNa
me=DeleteUserPolicy)||($.eventName=PutGroupPolicy)||($.eventName=PutRolePolic
y)||($.eventName=PutUserPolicy)||($.eventName=CreatePolicy)||($.eventName=Del
etePolicy)||($.eventName=CreatePolicyVersion)||($.eventName=DeletePolicyVersi
on)||($.eventName=AttachRolePolicy)||($.eventName=DetachRolePolicy)||($.event
Name=AttachUserPolicy)||($.eventName=DetachUserPolicy)||($.eventName=AttachGr
oupPolicy)||($.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 --threshold
1 --comparison-operator GreaterThanOrEqualToThreshold --evaluation-periods 1
--namespace 'CISBenchmark' --alarm-actions <sns_topic_arn>
```

References:

1. CCE-79189-7
2. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/receive-cloudtrail-log-files-from-multiple-regions.html
3. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html
4. https://docs.aws.amazon.com/sns/latest/dg/SubscribeTopic.html

Additional Information:

Configuring log metric filter and alarm on Multi-region (global) CloudTrail

- ensures that activities from all regions (used as well as unused) are monitored
- ensures that activities on all supported global services are monitored
- ensures that all management events across all regions are monitored

CIS Controls:

Version 7

16 Account Monitoring and Control
Account Monitoring and Control

4.5 Ensure a log metric filter and alarm exist for CloudTrail configuration changes (Automated)

Profile Applicability:

- Level 1

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 detecting changes to CloudTrail's configurations.

Rationale:

Monitoring changes to CloudTrail's configuration will help ensure sustained visibility to activities performed in the AWS account.

Audit:

Perform the following to ensure that there is at least one active multi-region CloudTrail with prescribed metric filters and alarms configured:

1. Identify the log group name configured for use with active multi-region CloudTrail:

- List all CloudTrails: `aws cloudtrail describe-trails`
- Identify Multi region Cloudtrails: `Trails with "IsMultiRegionTrail" set to true`
- From value associated with CloudWatchLogsLogGroupArn note `<cloudtrail_log_group_name>`

Example: for CloudWatchLogsLogGroupArn that looks like `arn:aws:logs:<region>:<aws_account_number>:log-group:NewGroup:*`, `<cloudtrail_log_group_name>` would be `NewGroup`

- Ensure Identified Multi region CloudTrail is active

`aws cloudtrail get-trail-status --name <Name of a Multi-region CloudTrail>` ensure `IsLogging` is set to `TRUE`

- Ensure identified Multi-region Cloudtrail captures all Management Events

```
aws cloudtrail get-event-selectors --trail-name <trailname shown in describe-trails>
```

Ensure there is at least one Event Selector for a Trail with `IncludeManagementEvents` set to `true` and `ReadWriteType` set to `All`

2. Get a list of all associated metric filters for this `<cloudtrail_log_group_name>`:

```
aws logs describe-metric-filters --log-group-name
"<cloudtrail_log_group_name>"
```

3. Ensure the output from the above command contains the following:

```
"filterPattern": "{ ($.eventName = CreateTrail) || ($.eventName =
UpdateTrail) || ($.eventName = DeleteTrail) || ($.eventName = StartLogging)
|| ($.eventName = StopLogging) }"
```

4. Note the `<cloudtrail_cfg_changes_metric>` value associated with the `filterPattern` found in step 3.
5. Get a list of CloudWatch alarms and filter on the `<cloudtrail_cfg_changes_metric>` captured in step 4.

```
aws cloudwatch describe-alarms --query 'MetricAlarms[?MetricName==
`<cloudtrail_cfg_changes_metric>`]'
```

6. Note the `AlarmActions` value - this will provide the SNS topic ARN value.
7. Ensure there is at least one active subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <sns_topic_arn>
```

at least one subscription should have "SubscriptionArn" with valid aws ARN.

```
Example of valid "SubscriptionArn":
"arn:aws:sns:<region>:<aws_account_number>:<SnsTopicName>:<SubscriptionID>"
```

Remediation:

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-transformations
metricName= `<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>
```

References:

1. CCE-79190-5
2. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/receive-cloudtrail-log-files-from-multiple-regions.html
3. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html
4. https://docs.aws.amazon.com/sns/latest/dg/SubscribeTopic.html

Additional Information:

Configuring log metric filter and alarm on Multi-region (global) CloudTrail

- ensures that activities from all regions (used as well as unused) are monitored
- ensures that activities on all supported global services are monitored
- ensures that all management events across all regions are monitored

CIS Controls:

Version 7

6 Maintenance, Monitoring and Analysis of Audit Logs
Maintenance, Monitoring and Analysis of Audit Logs

4.6 Ensure a log metric filter and alarm exist for AWS Management Console authentication failures (Automated)

Profile Applicability:

- Level 2

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 failed console authentication attempts.

Rationale:

Monitoring failed console logins may decrease lead time to detect an attempt to brute force a credential, which may provide an indicator, such as source IP, that can be used in other event correlation.

Audit:

Perform the following to ensure that there is at least one active multi-region CloudTrail with prescribed metric filters and alarms configured:

1. Identify the log group name configured for use with active multi-region CloudTrail:

- List all CloudTrails: `aws cloudtrail describe-trails`
- Identify Multi region Cloudtrails: `Trails with "IsMultiRegionTrail" set to true`
- From value associated with CloudWatchLogsLogGroupArn note `<cloudtrail_log_group_name>`

Example: for CloudWatchLogsLogGroupArn that looks like `arn:aws:logs:<region>:<aws_account_number>:log-group:NewGroup:*`, `<cloudtrail_log_group_name>` would be `NewGroup`

- Ensure Identified Multi region CloudTrail is active

`aws cloudtrail get-trail-status --name <Name of a Multi-region CloudTrail>` ensure `IsLogging` is set to `TRUE`

- Ensure identified Multi-region Cloudtrail captures all Management Events

```
aws cloudtrail get-event-selectors --trail-name <trailname shown in describe-trails>
```

Ensure there is at least one Event Selector for a Trail with `IncludeManagementEvents` set to `true` and `ReadWriteType` set to `All`

2. Get a list of all associated metric filters for this `<cloudtrail_log_group_name>`:

```
aws logs describe-metric-filters --log-group-name
"<cloudtrail_log_group_name>"
```

3. Ensure the output from the above command contains the following:

```
"filterPattern": "{ ($.eventName = ConsoleLogin) && ($.errorMessage = "Failed
authentication") }"
```

4. Note the `<console_signin_failure_metric>` value associated with the `filterPattern` found in step 3.
5. Get a list of CloudWatch alarms and filter on the `<console_signin_failure_metric>` captured in step 4.

```
aws cloudwatch describe-alarms --query 'MetricAlarms[?MetricName==
`<console_signin_failure_metric>`]'
```

6. Note the `AlarmActions` value - this will provide the SNS topic ARN value.
7. Ensure there is at least one active subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <sns_topic_arn>
```

at least one subscription should have "SubscriptionArn" with valid aws ARN.

```
Example of valid "SubscriptionArn":
"arn:aws:sns:<region>:<aws_account_number>:<SnsTopicName>:<SubscriptionID>"
```

Remediation:

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-transformations
metricName= `<console_signin_failure_metric>`
,metricNamespace='CISBenchmark',metricValue=1 --filter-pattern '{
($.eventName = ConsoleLogin) && ($.errorMessage = "Failed authentication") }'
```

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>
```

References:

1. CCE-79191-3
2. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/receive-cloudtrail-log-files-from-multiple-regions.html
3. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html
4. https://docs.aws.amazon.com/sns/latest/dg/SubscribeTopic.html

Additional Information:

Configuring log metric filter and alarm on Multi-region (global) CloudTrail

- ensures that activities from all regions (used as well as unused) are monitored
- ensures that activities on all supported global services are monitored
- ensures that all management events across all regions are monitored

CIS Controls:

Version 7

16 Account Monitoring and Control
Account Monitoring and Control

4.7 Ensure a log metric filter and alarm exist for disabling or scheduled deletion of customer created CMKs (Automated)

Profile Applicability:

- Level 2

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 customer created CMKs which have changed state to disabled or scheduled deletion.

Rationale:

Data encrypted with disabled or deleted keys will no longer be accessible.

Audit:

Perform the following to ensure that there is at least one active multi-region CloudTrail with prescribed metric filters and alarms configured:

1. Identify the log group name configured for use with active multi-region CloudTrail:

- List all CloudTrails: `aws cloudtrail describe-trails`
- Identify Multi region Cloudtrails: `Trails with "IsMultiRegionTrail" set to true`
- From value associated with CloudWatchLogsLogGroupArn note `<cloudtrail_log_group_name>`

Example: for CloudWatchLogsLogGroupArn that looks like `arn:aws:logs:<region>:<aws_account_number>:log-group:NewGroup:*`, `<cloudtrail_log_group_name>` would be `NewGroup`

- Ensure Identified Multi region CloudTrail is active

`aws cloudtrail get-trail-status --name <Name of a Multi-region CloudTrail>` ensure `IsLogging` is set to `TRUE`

- Ensure identified Multi-region Cloudtrail captures all Management Events

```
aws cloudtrail get-event-selectors --trail-name <trailname shown in describe-trails>
```

Ensure there is at least one Event Selector for a Trail with `IncludeManagementEvents` set to `true` and `ReadWriteType` set to `All`

2. Get a list of all associated metric filters for this `<cloudtrail_log_group_name>`:

```
aws logs describe-metric-filters --log-group-name
"<cloudtrail_log_group_name>"
```

3. Ensure the output from the above command contains the following:

```
"filterPattern": "{($.eventSource = kms.amazonaws.com) &&
(($.eventName=DisableKey)||($.eventName=ScheduleKeyDeletion)) }"
```

4. Note the `<disable_or_delete_cmk_changes_metric>` value associated with the `filterPattern` found in step 3.
5. Get a list of CloudWatch alarms and filter on the `<disable_or_delete_cmk_changes_metric>` captured in step 4.

```
aws cloudwatch describe-alarms --query 'MetricAlarms[?MetricName==
`<disable_or_delete_cmk_changes_metric>`]'
```

6. Note the `AlarmActions` value - this will provide the SNS topic ARN value.
7. Ensure there is at least one active subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <sns_topic_arn>
```

at least one subscription should have "SubscriptionArn" with valid aws ARN.

```
Example of valid "SubscriptionArn":
"arn:aws:sns:<region>:<aws_account_number>:<SnsTopicName>:<SubscriptionID>"
```

Remediation:

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>`  --metric-
transformations metricName= `<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 --evaluation-
periods 1 --namespace 'CISBenchmark' --alarm-actions <sns_topic_arn>
```

References:

1. CCE-79192-1
2. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html
3. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/receive-cloudtrail-log-files-from-multiple-regions.html
4. https://docs.aws.amazon.com/sns/latest/dg/SubscribeTopic.html

Additional Information:

Configuring log metric filter and alarm on Multi-region (global) CloudTrail

- ensures that activities from all regions (used as well as unused) are monitored
- ensures that activities on all supported global services are monitored
- ensures that all management events across all regions are monitored

CIS Controls:

Version 7

16 Account Monitoring and Control
Account Monitoring and Control

4.8 Ensure a log metric filter and alarm exist for S3 bucket policy changes (Automated)

Profile Applicability:

- Level 1

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 changes to S3 bucket policies.

Rationale:

Monitoring changes to S3 bucket policies may reduce time to detect and correct permissive policies on sensitive S3 buckets.

Audit:

Perform the following to ensure that there is at least one active multi-region CloudTrail with prescribed metric filters and alarms configured:

1. Identify the log group name configured for use with active multi-region CloudTrail:

- List all CloudTrails: `aws cloudtrail describe-trails`
- Identify Multi region Cloudtrails: `Trails with "IsMultiRegionTrail" set to true`
- From value associated with CloudWatchLogsLogGroupArn note `<cloudtrail_log_group_name>`

Example: for CloudWatchLogsLogGroupArn that looks like `arn:aws:logs:<region>:<aws_account_number>:log-group:NewGroup:*`, `<cloudtrail_log_group_name>` would be `NewGroup`

- Ensure Identified Multi region CloudTrail is active

`aws cloudtrail get-trail-status --name <Name of a Multi-region CloudTrail>` ensure `IsLogging` is set to `TRUE`

- Ensure identified Multi-region Cloudtrail captures all Management Events

```
aws cloudtrail get-event-selectors --trail-name <trailname shown in describe-trails>
```

Ensure there is at least one Event Selector for a Trail with `IncludeManagementEvents` set to `true` and `ReadWriteType` set to `All`

2. Get a list of all associated metric filters for this `<cloudtrail_log_group_name>`:

```
aws logs describe-metric-filters --log-group-name
"<cloudtrail_log_group_name>"
```

3. Ensure the output from the above command contains the following:

```
"filterPattern": "{ ($.eventSource = s3.amazonaws.com) && (($.eventName =
PutBucketAcl) || ($.eventName = PutBucketPolicy) || ($.eventName =
PutBucketCors) || ($.eventName = PutBucketLifecycle) || ($.eventName =
PutBucketReplication) || ($.eventName = DeleteBucketPolicy) || ($.eventName =
DeleteBucketCors) || ($.eventName = DeleteBucketLifecycle) || ($.eventName =
DeleteBucketReplication)) }"
```

4. Note the `<s3_bucket_policy_changes_metric>` value associated with the `filterPattern` found in step 3.
5. Get a list of CloudWatch alarms and filter on the `<s3_bucket_policy_changes_metric>` captured in step 4.

```
aws cloudwatch describe-alarms --query 'MetricAlarms[?MetricName==
`<s3_bucket_policy_changes_metric>`]'
```

6. Note the `AlarmActions` value - this will provide the SNS topic ARN value.
7. Ensure there is at least one active subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <sns_topic_arn>
```

at least one subscription should have "SubscriptionArn" with valid aws ARN.

```
Example of valid "SubscriptionArn":
"arn:aws:sns:<region>:<aws_account_number>:<SnsTopicName>:<SubscriptionID>"
```

Remediation:

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-transformations
metricName= `<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>`  --statistic Sum --period 300 --threshold
1 --comparison-operator GreaterThanOrEqualToThreshold --evaluation-periods 1
--namespace 'CISBenchmark' --alarm-actions <sns_topic_arn>
```

References:

1. CCE-79193-9
2. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html
3. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/receive-cloudtrail-log-files-from-multiple-regions.html
4. https://docs.aws.amazon.com/sns/latest/dg/SubscribeTopic.html

Additional Information:

Configuring log metric filter and alarm on Multi-region (global) CloudTrail

- ensures that activities from all regions (used as well as unused) are monitored
- ensures that activities on all supported global services are monitored
- ensures that all management events across all regions are monitored

CIS Controls:

Version 7

6.2 Activate audit logging
Ensure that local logging has been enabled on all systems and networking devices.

14 Controlled Access Based on the Need to Know
Controlled Access Based on the Need to Know

4.9 Ensure a log metric filter and alarm exist for AWS Config configuration changes (Automated)

Profile Applicability:

- Level 2

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 detecting changes to CloudTrail's configurations.

Rationale:

Monitoring changes to AWS Config configuration will help ensure sustained visibility of configuration items within the AWS account.

Audit:

Perform the following to ensure that there is at least one active multi-region CloudTrail with prescribed metric filters and alarms configured:

1. Identify the log group name configured for use with active multi-region CloudTrail:

- List all CloudTrails: `aws cloudtrail describe-trails`
- Identify Multi region Cloudtrails: `Trails with "IsMultiRegionTrail" set to true`
- From value associated with CloudWatchLogsLogGroupArn note `<cloudtrail_log_group_name>`

Example: for CloudWatchLogsLogGroupArn that looks like `arn:aws:logs:<region>:<aws_account_number>:log-group:NewGroup:*`, `<cloudtrail_log_group_name>` would be `NewGroup`

- Ensure Identified Multi region CloudTrail is active

`aws cloudtrail get-trail-status --name <Name of a Multi-region CloudTrail>` ensure `IsLogging` is set to `TRUE`

- Ensure identified Multi-region Cloudtrail captures all Management Events

```
aws cloudtrail get-event-selectors --trail-name <trailname shown in describe-trails>
```

Ensure there is at least one Event Selector for a Trail with `IncludeManagementEvents` set to `true` and `ReadWriteType` set to `All`

2. Get a list of all associated metric filters for this `<cloudtrail_log_group_name>`:

```
aws logs describe-metric-filters --log-group-name
"<cloudtrail_log_group_name>"
```

3. Ensure the output from the above command contains the following:

```
"filterPattern": "{ ($.eventSource = config.amazonaws.com) &&
(($.eventName=StopConfigurationRecorder)||($.eventName=DeleteDeliveryChannel)
||($.eventName=PutDeliveryChannel)||($.eventName=PutConfigurationRecorder))
}"
```

4. Note the `<aws_config_changes_metric>` value associated with the `filterPattern` found in step 3.
5. Get a list of CloudWatch alarms and filter on the `<aws_config_changes_metric>` captured in step 4.

```
aws cloudwatch describe-alarms --query 'MetricAlarms[?MetricName==
`<aws_config_changes_metric>`]'
```

6. Note the `AlarmActions` value - this will provide the SNS topic ARN value.
7. Ensure there is at least one active subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <sns_topic_arn>
```

at least one subscription should have "SubscriptionArn" with valid aws ARN.

```
Example of valid "SubscriptionArn":
"arn:aws:sns:<region>:<aws_account_number>:<SnsTopicName>:<SubscriptionID>"
```

Remediation:

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-transformations
metricName= `<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 --evaluation-
periods 1 --namespace 'CISBenchmark' --alarm-actions <sns_topic_arn>
```

References:

1. CCE-79194-7
2. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html
3. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/receive-cloudtrail-log-files-from-multiple-regions.html
4. https://docs.aws.amazon.com/sns/latest/dg/SubscribeTopic.html

Additional Information:

Configuring log metric filter and alarm on Multi-region (global) CloudTrail

- ensures that activities from all regions (used as well as unused) are monitored
- ensures that activities on all supported global services are monitored
- ensures that all management events across all regions are monitored

CIS Controls:

Version 7

1.4 Maintain Detailed Asset Inventory

Maintain an accurate and up-to-date inventory of all technology assets with the potential to store or process information. This inventory shall include all hardware assets, whether connected to the organization's network or not.

11.2 Document Traffic Configuration Rules

All configuration rules that allow traffic to flow through network devices should be documented in a configuration management system with a specific business reason for each rule, a specific individual’s name responsible for that business need, and an expected duration of the need.

16.1 Maintain an Inventory of Authentication Systems

Maintain an inventory of each of the organization's authentication systems, including those located onsite or at a remote service provider.

4.10 Ensure a log metric filter and alarm exist for security group changes (Automated)

Profile Applicability:

- Level 2

Description:

Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs 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.

Rationale:

Monitoring changes to security group will help ensure that resources and services are not unintentionally exposed.

Audit:

Perform the following to ensure that there is at least one active multi-region CloudTrail with prescribed metric filters and alarms configured:

1. Identify the log group name configured for use with active multi-region CloudTrail:

- List all CloudTrails: `aws cloudtrail describe-trails`
- Identify Multi region Cloudtrails: `Trails with "IsMultiRegionTrail" set to true`
- From value associated with CloudWatchLogsLogGroupArn note `<cloudtrail_log_group_name>`

Example: for CloudWatchLogsLogGroupArn that looks like `arn:aws:logs:<region>:<aws_account_number>:log-group:NewGroup:*`, `<cloudtrail_log_group_name>` would be `NewGroup`

- Ensure Identified Multi region CloudTrail is active

`aws cloudtrail get-trail-status --name <Name of a Multi-region CloudTrail>` ensure `IsLogging` is set to `TRUE`

- Ensure identified Multi-region Cloudtrail captures all Management Events

```
aws cloudtrail get-event-selectors --trail-name <trailname shown in describe-trails>
```

Ensure there is at least one Event Selector for a Trail with `IncludeManagementEvents` set to `true` and `ReadWriteType` set to `All`

2. Get a list of all associated metric filters for this `<cloudtrail_log_group_name>`:

```
aws logs describe-metric-filters --log-group-name
"<cloudtrail_log_group_name>"
```

3. Ensure the output from the above command contains the following:

```
"filterPattern": "{ ($.eventName = AuthorizeSecurityGroupIngress) ||
($.eventName = AuthorizeSecurityGroupEgress) || ($.eventName =
RevokeSecurityGroupIngress) || ($.eventName = RevokeSecurityGroupEgress) ||
($.eventName = CreateSecurityGroup) || ($.eventName = DeleteSecurityGroup) }"
```

4. Note the `<security_group_changes_metric>` value associated with the `filterPattern` found in step 3.
5. Get a list of CloudWatch alarms and filter on the `<security_group_changes_metric>` captured in step 4.

```
aws cloudwatch describe-alarms --query "MetricAlarms[?MetricName==
'<security_group_changes_metric>']"
```

6. Note the `AlarmActions` value - this will provide the SNS topic ARN value.
7. Ensure there is at least one active subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <sns_topic_arn>
```

at least one subscription should have "SubscriptionArn" with valid aws ARN.

```
Example of valid "SubscriptionArn":
"arn:aws:sns:<region>:<aws_account_number>:<SnsTopicName>:<SubscriptionID>"
```

Remediation:

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-transformations
metricName= "<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>"  --statistic Sum --period 300 --threshold 1
--comparison-operator GreaterThanOrEqualToThreshold --evaluation-periods 1 --
namespace "CISBenchmark" --alarm-actions "<sns_topic_arn>"
```

References:

1. CCE-79195-4
2. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/receive-cloudtrail-log-files-from-multiple-regions.html
3. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html
4. https://docs.aws.amazon.com/sns/latest/dg/SubscribeTopic.html

Additional Information:

Configuring log metric filter and alarm on Multi-region (global) CloudTrail

- ensures that activities from all regions (used as well as unused) are monitored
- ensures that activities on all supported global services are monitored
- ensures that all management events across all regions are monitored

CIS Controls:

Version 7

6.2 Activate audit logging

Ensure that local logging has been enabled on all systems and networking devices.

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

4.11 Ensure a log metric filter and alarm exist for changes to Network Access Control Lists (NACL) (Automated)

Profile Applicability:

- Level 2

Description:

Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs 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.

Rationale:

Monitoring changes to NACLs will help ensure that AWS resources and services are not unintentionally exposed.

Audit:

Perform the following to ensure that there is at least one active multi-region CloudTrail with prescribed metric filters and alarms configured:

1. Identify the log group name configured for use with active multi-region CloudTrail:

- List all CloudTrails: `aws cloudtrail describe-trails`
- Identify Multi region Cloudtrails: `Trails with "IsMultiRegionTrail" set to true`
- From value associated with CloudWatchLogsLogGroupArn note `<cloudtrail_log_group_name>`

Example: for CloudWatchLogsLogGroupArn that looks like `arn:aws:logs:<region>:<aws_account_number>:log-group:NewGroup:*`, `<cloudtrail_log_group_name>` would be `NewGroup`

- Ensure Identified Multi region CloudTrail is active

`aws cloudtrail get-trail-status --name <Name of a Multi-region CloudTrail>` ensure `IsLogging` is set to `TRUE`

- Ensure identified Multi-region Cloudtrail captures all Management Events

```
aws cloudtrail get-event-selectors --trail-name <trailname shown in describe-trails>
```

Ensure there is at least one Event Selector for a Trail with `IncludeManagementEvents` set to `true` and `ReadWriteType` set to `All`

2. Get a list of all associated metric filters for this `<cloudtrail_log_group_name>`:

```
aws logs describe-metric-filters --log-group-name
"<cloudtrail_log_group_name>"
```

3. Ensure the output from the above command contains the following:

```
"filterPattern": "{ ($.eventName = CreateNetworkAcl) || ($.eventName =
CreateNetworkAclEntry) || ($.eventName = DeleteNetworkAcl) || ($.eventName =
DeleteNetworkAclEntry) || ($.eventName = ReplaceNetworkAclEntry) ||
($.eventName = ReplaceNetworkAclAssociation) }"
```

4. Note the `<nacl_changes_metric>` value associated with the `filterPattern` found in step 3.
5. Get a list of CloudWatch alarms and filter on the `<nacl_changes_metric>` captured in step 4.

```
aws cloudwatch describe-alarms --query 'MetricAlarms[?MetricName==
`<nacl_changes_metric>`]'
```

6. Note the `AlarmActions` value - this will provide the SNS topic ARN value.
7. Ensure there is at least one active subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <sns_topic_arn>
```

at least one subscription should have "SubscriptionArn" with valid aws ARN.

```
Example of valid "SubscriptionArn":
"arn:aws:sns:<region>:<aws_account_number>:<SnsTopicName>:<SubscriptionID>"
```

Remediation:

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 --evaluation-
periods 1 --namespace 'CISBenchmark' --alarm-actions <sns_topic_arn>
```

References:

1. CCE-79196-2
2. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/receive-cloudtrail-log-files-from-multiple-regions.html
3. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html
4. https://docs.aws.amazon.com/sns/latest/dg/SubscribeTopic.html

Additional Information:

Configuring log metric filter and alarm on Multi-region (global) CloudTrail

- ensures that activities from all regions (used as well as unused) are monitored
- ensures that activities on all supported global services are monitored
- ensures that all management events across all regions are monitored

CIS Controls:

Version 7

11.3 Use Automated Tools to Verify Standard Device Configurations and Detect Changes

Compare all network device configuration against approved security configurations defined for each network device in use and alert when any deviations are discovered.

4.12 Ensure a log metric filter and alarm exist for changes to network gateways (Automated)

Profile Applicability:

- Level 1

Description:

Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs 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 gateways.

Rationale:

Monitoring changes to network gateways will help ensure that all ingress/egress traffic traverses the VPC border via a controlled path.

Audit:

Perform the following to ensure that there is at least one active multi-region CloudTrail with prescribed metric filters and alarms configured:

1. Identify the log group name configured for use with active multi-region CloudTrail:

- List all CloudTrails: `aws cloudtrail describe-trails`
- Identify Multi region Cloudtrails: `Trails with "IsMultiRegionTrail" set to true`
- From value associated with CloudWatchLogsLogGroupArn note `<cloudtrail_log_group_name>`

Example: for CloudWatchLogsLogGroupArn that looks like `arn:aws:logs:<region>:<aws_account_number>:log-group:NewGroup:*`, `<cloudtrail_log_group_name>` would be `NewGroup`

- Ensure Identified Multi region CloudTrail is active

`aws cloudtrail get-trail-status --name <Name of a Multi-region CloudTrail>` ensure `IsLogging` is set to `TRUE`

- Ensure identified Multi-region Cloudtrail captures all Management Events

```
aws cloudtrail get-event-selectors --trail-name <trailname shown in describe-trails>
```

Ensure there is at least one Event Selector for a Trail with `IncludeManagementEvents` set to `true` and `ReadWriteType` set to `All`

2. Get a list of all associated metric filters for this `<cloudtrail_log_group_name>`:

```
aws logs describe-metric-filters --log-group-name
"<cloudtrail_log_group_name>"
```

3. Ensure the output from the above command contains the following:

```
"filterPattern": "{ ($.eventName = CreateCustomerGateway) || ($.eventName =
DeleteCustomerGateway) || ($.eventName = AttachInternetGateway) ||
($.eventName = CreateInternetGateway) || ($.eventName =
DeleteInternetGateway) || ($.eventName = DetachInternetGateway) }"
```

4. Note the `<network_gw_changes_metric>` value associated with the `filterPattern` found in step 3.
5. Get a list of CloudWatch alarms and filter on the `<network_gw_changes_metric>` captured in step 4.

```
aws cloudwatch describe-alarms --query 'MetricAlarms[?MetricName==
`<network_gw_changes_metric>`]'
```

6. Note the `AlarmActions` value - this will provide the SNS topic ARN value.
7. Ensure there is at least one active subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <sns_topic_arn>
```

at least one subscription should have "SubscriptionArn" with valid aws ARN.

```
Example of valid "SubscriptionArn":
"arn:aws:sns:<region>:<aws_account_number>:<SnsTopicName>:<SubscriptionID>"
```

Remediation:

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 Sum --period 300 --
threshold 1 --comparison-operator GreaterThanOrEqualToThreshold --evaluation-
periods 1 --namespace 'CISBenchmark' --alarm-actions <sns_topic_arn>
```

References:

1. CCE-79197-0
2. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/receive-cloudtrail-log-files-from-multiple-regions.html
3. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html
4. https://docs.aws.amazon.com/sns/latest/dg/SubscribeTopic.html

Additional Information:

Configuring log metric filter and alarm on Multi-region (global) CloudTrail

- ensures that activities from all regions (used as well as unused) are monitored
- ensures that activities on all supported global services are monitored
- ensures that all management events across all regions are monitored

CIS Controls:

Version 7

6.2 Activate audit logging

Ensure that local logging has been enabled on all systems and networking devices.

11.3 Use Automated Tools to Verify Standard Device Configurations and Detect Changes

Compare all network device configuration against approved security configurations defined for each network device in use and alert when any deviations are discovered.

4.13 Ensure a log metric filter and alarm exist for route table changes (Automated)

Profile Applicability:

- Level 1

Description:

Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs 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.

Rationale:

Monitoring changes to route tables will help ensure that all VPC traffic flows through an expected path.

Audit:

Perform the following to ensure that there is at least one active multi-region CloudTrail with prescribed metric filters and alarms configured:

1. Identify the log group name configured for use with active multi-region CloudTrail:

- List all CloudTrails: `aws cloudtrail describe-trails`
- Identify Multi region Cloudtrails: `Trails with "IsMultiRegionTrail" set to true`
- From value associated with CloudWatchLogsLogGroupArn note `<cloudtrail_log_group_name>`

Example: for CloudWatchLogsLogGroupArn that looks like `arn:aws:logs:<region>:<aws_account_number>:log-group:NewGroup:*`, `<cloudtrail_log_group_name>` would be `NewGroup`

- Ensure Identified Multi region CloudTrail is active

`aws cloudtrail get-trail-status --name <Name of a Multi-region CloudTrail>` ensure `IsLogging` is set to `TRUE`

- Ensure identified Multi-region Cloudtrail captures all Management Events

```
aws cloudtrail get-event-selectors --trail-name <trailname shown in describe-trails>
```

Ensure there is at least one Event Selector for a Trail with `IncludeManagementEvents` set to `true` and `ReadWriteType` set to `All`

2. Get a list of all associated metric filters for this `<cloudtrail_log_group_name>`:

```
aws logs describe-metric-filters --log-group-name
"<cloudtrail_log_group_name>"
```

3. Ensure the output from the above command contains the following:

```
"filterPattern": "{ ($.eventName = CreateRoute) || ($.eventName =
CreateRouteTable) || ($.eventName = ReplaceRoute) || ($.eventName =
ReplaceRouteTableAssociation) || ($.eventName = DeleteRouteTable) ||
($.eventName = DeleteRoute) || ($.eventName = DisassociateRouteTable) }"
```

4. Note the `<route_table_changes_metric>` value associated with the `filterPattern` found in step 3.
5. Get a list of CloudWatch alarms and filter on the `<route_table_changes_metric>` captured in step 4.

```
aws cloudwatch describe-alarms --query 'MetricAlarms[?MetricName==
`<route_table_changes_metric>`]'
```

6. Note the `AlarmActions` value - this will provide the SNS topic ARN value.
7. Ensure there is at least one active subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <sns_topic_arn>
```

at least one subscription should have "SubscriptionArn" with valid aws ARN.

```
Example of valid "SubscriptionArn":
"arn:aws:sns:<region>:<aws_account_number>:<SnsTopicName>:<SubscriptionID>"
```

Remediation:

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-transformations
metricName= `<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>
```

References:

1. CCE-79198-8
2. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/receive-cloudtrail-log-files-from-multiple-regions.html
3. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html
4. https://docs.aws.amazon.com/sns/latest/dg/SubscribeTopic.html

Additional Information:

Configuring log metric filter and alarm on Multi-region (global) CloudTrail

- ensures that activities from all regions (used as well as unused) are monitored
- ensures that activities on all supported global services are monitored
- ensures that all management events across all regions are monitored

CIS Controls:

Version 7

6.2 Activate audit logging

Ensure that local logging has been enabled on all systems and networking devices.

11.3 Use Automated Tools to Verify Standard Device Configurations and Detect Changes

Compare all network device configuration against approved security configurations defined for each network device in use and alert when any deviations are discovered.

4.14 Ensure a log metric filter and alarm exist for VPC changes (Automated)

Profile Applicability:

- Level 1

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 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.

Rationale:

Monitoring changes to VPC will help ensure VPC traffic flow is not getting impacted.

Audit:

Perform the following to ensure that there is at least one active multi-region CloudTrail with prescribed metric filters and alarms configured:

1. Identify the log group name configured for use with active multi-region CloudTrail:

- List all CloudTrails: `aws cloudtrail describe-trails`
- Identify Multi region Cloudtrails: `Trails with "IsMultiRegionTrail" set to true`
- From value associated with CloudWatchLogsLogGroupArn note `<cloudtrail_log_group_name>`

Example: for CloudWatchLogsLogGroupArn that looks like `arn:aws:logs:<region>:<aws_account_number>:log-group:NewGroup:*`, `<cloudtrail_log_group_name>` would be `NewGroup`

- Ensure Identified Multi region CloudTrail is active

`aws cloudtrail get-trail-status --name <Name of a Multi-region CloudTrail>` ensure `IsLogging` is set to `TRUE`

- Ensure identified Multi-region Cloudtrail captures all Management Events

```
aws cloudtrail get-event-selectors --trail-name <trailname shown in describe-trails>
```

Ensure there is at least one Event Selector for a Trail with `IncludeManagementEvents` set to `true` and `ReadWriteType` set to `All`

2. Get a list of all associated metric filters for this `<cloudtrail_log_group_name>`:

```
aws logs describe-metric-filters --log-group-name
"<cloudtrail_log_group_name>"
```

3. Ensure the output from the above command contains the following:

```
"filterPattern": "{ ($.eventName = CreateVpc) || ($.eventName = DeleteVpc) ||
($.eventName = ModifyVpcAttribute) || ($.eventName =
AcceptVpcPeeringConnection) || ($.eventName = CreateVpcPeeringConnection) ||
($.eventName = DeleteVpcPeeringConnection) || ($.eventName =
RejectVpcPeeringConnection) || ($.eventName = AttachClassicLinkVpc) ||
($.eventName = DetachClassicLinkVpc) || ($.eventName = DisableVpcClassicLink)
|| ($.eventName = EnableVpcClassicLink) }"
```

4. Note the `<vpc_changes_metric>` value associated with the `filterPattern` found in step 3.
5. Get a list of CloudWatch alarms and filter on the `<vpc_changes_metric>` captured in step 4.

```
aws cloudwatch describe-alarms --query 'MetricAlarms[?MetricName==
`<vpc_changes_metric>`]'
```

6. Note the `AlarmActions` value - this will provide the SNS topic ARN value.
7. Ensure there is at least one active subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <sns_topic_arn>
```

at least one subscription should have "SubscriptionArn" with valid aws ARN.

```
Example of valid "SubscriptionArn":
"arn:aws:sns:<region>:<aws_account_number>:<SnsTopicName>:<SubscriptionID>"
```

Remediation:

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>
```

References:

1. CCE-79199-6
2. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/receive-cloudtrail-log-files-from-multiple-regions.html
3. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html
4. https://docs.aws.amazon.com/sns/latest/dg/SubscribeTopic.html

Additional Information:

Configuring log metric filter and alarm on Multi-region (global) CloudTrail

- ensures that activities from all regions (used as well as unused) are monitored
- ensures that activities on all supported global services are monitored
- ensures that all management events across all regions are monitored

CIS Controls:

Version 7

5.5 Implement Automated Configuration Monitoring Systems

Utilize a Security Content Automation Protocol (SCAP) compliant configuration monitoring system to verify all security configuration elements, catalog approved exceptions, and alert when unauthorized changes occur.

4.15 Ensure a log metric filter and alarm exists for AWS Organizations changes (Automated)

Profile Applicability:

- Level 1

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.

Rationale:

Monitoring AWS Organizations changes can help you prevent any unwanted, accidental or intentional modifications that may lead to unauthorized access or other security breaches. This monitoring technique helps you to ensure that any unexpected changes performed within your AWS Organizations can be investigated and any unwanted changes can be rolled back.

Audit:

1. Perform the following to ensure that there is at least one active multi-region CloudTrail with prescribed metric filters and alarms configured:

- Identify the log group name configured for use with active multi-region CloudTrail:
- List all CloudTrails:

```
aws cloudtrail describe-trails
```

- Identify Multi region Cloudtrails, Trails with `"IsMultiRegionTrail"` set to true
- From value associated with CloudWatchLogsLogGroupArn note <cloudtrail_log_group_name>
  Example: for CloudWatchLogsLogGroupArn that looks like arn:aws:logs::<aws_account_number>:log-group:NewGroup:*, <cloudtrail_log_group_name> would be NewGroup
- Ensure Identified Multi region CloudTrail is active:

```
aws cloudtrail get-trail-status --name <Name of a Multi-region CloudTrail>
```

Ensure `IsLogging` is set to `TRUE`

- Ensure identified Multi-region Cloudtrail captures all Management Events:

```
aws cloudtrail get-event-selectors --trail-name <trailname shown in describe-
trails>
```

- Ensure there is at least one Event Selector for a Trail with `IncludeManagementEvents` set to true and `ReadWriteType` set to `All`.

2. Get a list of all associated metric filters for this <cloudtrail_log_group_name>:

```
aws logs describe-metric-filters --log-group-name
"<cloudtrail_log_group_name>"
```

3. Ensure the output from the above command contains the following:

```
"filterPattern": "{ ($.eventSource = organizations.amazonaws.com) &&
(($.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")) }"
```

4. Note the `<organizations_changes>` value associated with the filterPattern found in step 3.
5. Get a list of CloudWatch alarms and filter on the `<organizations_changes>` captured in step 4:

```
aws cloudwatch describe-alarms --query 'MetricAlarms[?MetricName==
`<organizations_changes>`]'
```

6. Note the AlarmActions value - this will provide the SNS topic ARN value.
7. Ensure there is at least one active subscriber to the SNS topic:

```
aws sns list-subscriptions-by-topic --topic-arn <sns_topic_arn>
```

at least one subscription should have "SubscriptionArn" with valid aws ARN.
Example of valid "SubscriptionArn":

```
"arn:aws:sns:<region>:<aws_account_number>:<SnsTopicName>:<SubscriptionID>"
```

Remediation:

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 --period 300 --
threshold 1 --comparison-operator GreaterThanOrEqualToThreshold --evaluation-
periods 1 --namespace 'CISBenchmark' --alarm-actions <sns_topic_arn>
```

References:

1. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html
2. https://docs.aws.amazon.com/organizations/latest/userguide/orgs_security_incident-response.html

CIS Controls:

Version 7

6.2 Activate audit logging

Ensure that local logging has been enabled on all systems and networking devices.

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

5 Networking

This section contains recommendations for configuring security-related aspects of the default Virtual Private Cloud (VPC)

5.1 Ensure no Network ACLs allow ingress from 0.0.0.0/0 to remote server administration ports (Automated)

Profile Applicability:

- Level 1

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`.

Rationale:

Public access to remote server administration ports, such as 22 and 3389, increases resource attack surface and unnecessarily raises the risk of resource compromise.

Audit:

From Console:
Perform the following to determine if the account is configured as prescribed:

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, perform the following:

    - Select the network ACL
    - Click the `Inbound Rules` tab
    - Ensure no rule exists that has a port range that includes port `22`, `3389`, or other remote server administration ports for your environment and has a `Source` of `0.0.0.0/0` and shows `ALLOW`

Note: A Port value of `ALL` or a port range such as `0-1024` are inclusive of port `22`, `3389`, and other remote server administration ports.

Remediation:

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`

References:

1. https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html
2. https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Security.html#VPC_Security_Comparison

CIS Controls:

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running

Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

12.4 Deny Communication over Unauthorized Ports

Deny communication over unauthorized TCP or UDP ports or application traffic to ensure that only authorized protocols are allowed to cross the network boundary in or out of the network at each of the organization's network boundaries.

5.2 Ensure no security groups allow ingress from 0.0.0.0/0 to remote server administration ports (Automated)

Profile Applicability:

- Level 1

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`.

Rationale:

Public access to remote server administration ports, such as 22 and 3389, increases resource attack surface and unnecessarily raises the risk of resource compromise.

Impact:

When updating an existing environment, ensure that administrators have access to remote server administration ports through another mechanism before removing access by deleting the 0.0.0.0/0 inbound rule.

Audit:

Perform the following to determine if the account is configured as prescribed:

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. Ensure no rule exists that has a port range that includes port `22`, `3389`, or other remote server administration ports for your environment and has a `Source` of `0.0.0.0/0`

Note: A Port value of `ALL` or a port range such as `0-1024` are inclusive of port `22`, `3389`, and other remote server administration ports.

Remediation:

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`

References:

1. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-security-groups.html#deleting-security-group-rule

CIS Controls:

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running

Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

12.4 Deny Communication over Unauthorized Ports

Deny communication over unauthorized TCP or UDP ports or application traffic to ensure that only authorized protocols are allowed to cross the network boundary in or out of the network at each of the organization's network boundaries.

5.3 Ensure the default security group of every VPC restricts all traffic (Automated)

Profile Applicability:

- Level 2

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.

Rationale:

Configuring all VPC default security groups to restrict all traffic will encourage least privilege security group development and mindful placement of AWS resources into security groups which will in-turn reduce the exposure of those resources.

Impact:

Implementing this recommendation in an existing VPC containing operating resources requires extremely careful migration planning as the default security groups are likely to be enabling many ports that are unknown. Enabling VPC flow logging (of accepts) in an existing environment that is known to be breach free will reveal the current pattern of ports being used for each instance to communicate successfully.

Audit:

Perform the following to determine if the account is configured as prescribed:
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. Ensure no rule exist
8. Click the `Outbound Rules` tab
9. Ensure no rules exist

Security Group Members

1. Login to the AWS Management Console at https://console.aws.amazon.com/vpc/home
2. Repeat the next steps for all default groups in all VPCs - including the default VPC in each AWS region:
3. In the left pane, click `Security Groups`
4. Copy the id of the default security group.
5. Change to the EC2 Management Console at https://console.aws.amazon.com/ec2/v2/home
6. In the filter column type 'Security Group ID : < security group id from #4 >'

Remediation:

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 inbound 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"

References:

1. CCE-79201-0
2. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html
3. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-security-groups.html#default-security-group

CIS Controls:

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

5.4 Ensure routing tables for VPC peering are "least access" (Manual)

Profile Applicability:

- Level 2

Description:

Once a VPC peering connection is established, routing tables must be updated to establish any connections between the peered VPCs. These routes can be as specific as desired - even peering a VPC to only a single host on the other side of the connection.

Rationale:

Being highly selective in peering routing tables is a very effective way of minimizing the impact of breach as resources outside of these routes are inaccessible to the peered VPC.

Audit:

Review routing tables of peered VPCs for whether they route all subnets of each VPC and whether that is necessary to accomplish the intended purposes for peering the VPCs.
From Command Line:

1. List all the route tables from a VPC and check if "GatewayId" is pointing to a *<peering_connection_id>* (e.g. pcx-1a2b3c4d) and if "DestinationCidrBlock" is as specific as desired.

```
aws ec2 describe-route-tables --filter "Name=vpc-id,Values=<vpc id>" --query
"RouteTables[*].{RouteTableId:RouteTableId, VpcId:VpcId, Routes:Routes,
AssociatedSubnets:Associations[*].SubnetId}"
```

Remediation:

Remove and add route table entries to ensure that the least number of subnets or hosts as is required to accomplish the purpose for peering are routable.
From Command Line:

1. For each *<route_table_id>* containing routes non compliant with your routing policy (which grants more than desired "least access"), delete the non compliant route:

```
aws ec2 delete-route --route-table-id <route_table_id> --destination-cidr-
block <non_compliant_destination_CIDR>
```

2. Create a new compliant route:

```
aws ec2 create-route --route-table-id <route_table_id> --destination-cidr-
block <compliant_destination_CIDR> --vpc-peering-connection-id
<peering_connection_id>
```

References:

1. https://docs.aws.amazon.com/AmazonVPC/latest/PeeringGuide/peering-configurations-partial-access.html
2. https://docs.aws.amazon.com/cli/latest/reference/ec2/create-vpc-peering-connection.html

Additional Information:

If an organization has AWS transit gateway implemented in their VPC architecture they should look to apply the recommendation above for "least access" routing architecture at the AWS transit gateway level in combination with what must be implemented at the standard VPC route table. More specifically, to route traffic between two or more VPCs via a transit gateway VPCs must have an attachment to a transit gateway route table as well as a route, therefore to avoid routing traffic between VPCs an attachment to the transit gateway route table should only be added where there is an intention to route traffic between the VPCs. As transit gateways are able to host multiple route tables it is possible to group VPCs by attaching them to a common route table.

CIS Controls:

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

Appendix: Summary Table

Control		Set Correctly	
		Yes	No
1	**Identity and Access Management**		
1.1	Maintain current contact details (Manual)	☐	☐
1.2	Ensure security contact information is registered (Manual)	☐	☐
1.3	Ensure security questions are registered in the AWS account (Manual)	☐	☐
1.4	Ensure no root user account access key exists (Automated)	☐	☐
1.5	Ensure MFA is enabled for the "root user" account (Automated)	☐	☐
1.6	Ensure hardware MFA is enabled for the "root user" account (Automated)	☐	☐
1.7	Eliminate use of the root user for administrative and daily tasks (Automated)	☐	☐
1.8	Ensure IAM password policy requires minimum length of 14 or greater (Automated)	☐	☐
1.9	Ensure IAM password policy prevents password reuse (Automated)	☐	☐
1.10	Ensure multi-factor authentication (MFA) is enabled for all IAM users that have a console password (Automated)	☐	☐
1.11	Do not setup access keys during initial user setup for all IAM users that have a console password (Manual)	☐	☐
1.12	Ensure credentials unused for 90 days or greater are disabled (Automated)	☐	☐
1.13	Ensure there is only one active access key available for any single IAM user (Automated)	☐	☐
1.14	Ensure access keys are rotated every 90 days or less (Automated)	☐	☐
1.15	Ensure IAM Users Receive Permissions Only Through Groups (Automated)	☐	☐
1.16	Ensure IAM policies that allow full "*:*" administrative privileges are not attached (Automated)	☐	☐
1.17	Ensure a support role has been created to manage incidents with AWS Support (Automated)	☐	☐
1.18	Ensure IAM instance roles are used for AWS resource access from instances (Manual)	☐	☐
1.19	Ensure that all the expired SSL/TLS certificates stored in AWS IAM are removed (Automated)	☐	☐
1.20	Ensure that S3 Buckets are configured with 'Block public access (bucket settings)' (Automated)	☐	☐

1.21	Ensure that IAM Access analyzer is enabled (Automated)	☐	☐
1.22	Ensure IAM users are managed centrally via identity federation or AWS Organizations for multi-account environments (Manual)	☐	☐
2	**Storage**		
2.1	**Simple Storage Service (S3)**		
2.1.1	Ensure all S3 buckets employ encryption-at-rest (Manual)	☐	☐
2.1.2	Ensure S3 Bucket Policy allows HTTPS requests (Manual)	☐	☐
2.2	**Elastic Compute Cloud (EC2)**		
2.2.1	Ensure EBS volume encryption is enabled (Manual)	☐	☐
3	**Logging**		
3.1	Ensure CloudTrail is enabled in all regions (Automated)	☐	☐
3.2	Ensure CloudTrail log file validation is enabled (Automated)	☐	☐
3.3	Ensure the S3 bucket used to store CloudTrail logs is not publicly accessible (Automated)	☐	☐
3.4	Ensure CloudTrail trails are integrated with CloudWatch Logs (Automated)	☐	☐
3.5	Ensure AWS Config is enabled in all regions (Automated)	☐	☐
3.6	Ensure S3 bucket access logging is enabled on the CloudTrail S3 bucket (Automated)	☐	☐
3.7	Ensure CloudTrail logs are encrypted at rest using KMS CMKs (Automated)	☐	☐
3.8	Ensure rotation for customer created CMKs is enabled (Automated)	☐	☐
3.9	Ensure VPC flow logging is enabled in all VPCs (Automated)	☐	☐
3.10	Ensure that Object-level logging for write events is enabled for S3 bucket (Automated)	☐	☐
3.11	Ensure that Object-level logging for read events is enabled for S3 bucket (Automated)	☐	☐
4	**Monitoring**		
4.1	Ensure a log metric filter and alarm exist for unauthorized API calls (Automated)	☐	☐
4.2	Ensure a log metric filter and alarm exist for Management Console sign-in without MFA (Automated)	☐	☐
4.3	Ensure a log metric filter and alarm exist for usage of "root" account (Automated)	☐	☐
4.4	Ensure a log metric filter and alarm exist for IAM policy changes (Automated)	☐	☐
4.5	Ensure a log metric filter and alarm exist for CloudTrail configuration changes (Automated)	☐	☐
4.6	Ensure a log metric filter and alarm exist for AWS Management Console authentication failures (Automated)	☐	☐

4.7	Ensure a log metric filter and alarm exist for disabling or scheduled deletion of customer created CMKs (Automated)	☐	☐
4.8	Ensure a log metric filter and alarm exist for S3 bucket policy changes (Automated)	☐	☐
4.9	Ensure a log metric filter and alarm exist for AWS Config configuration changes (Automated)	☐	☐
4.10	Ensure a log metric filter and alarm exist for security group changes (Automated)	☐	☐
4.11	Ensure a log metric filter and alarm exist for changes to Network Access Control Lists (NACL) (Automated)	☐	☐
4.12	Ensure a log metric filter and alarm exist for changes to network gateways (Automated)	☐	☐
4.13	Ensure a log metric filter and alarm exist for route table changes (Automated)	☐	☐
4.14	Ensure a log metric filter and alarm exist for VPC changes (Automated)	☐	☐
4.15	Ensure a log metric filter and alarm exists for AWS Organizations changes (Automated)	☐	☐
5	**Networking**		
5.1	Ensure no Network ACLs allow ingress from 0.0.0.0/0 to remote server administration ports (Automated)	☐	☐
5.2	Ensure no security groups allow ingress from 0.0.0.0/0 to remote server administration ports (Automated)	☐	☐
5.3	Ensure the default security group of every VPC restricts all traffic (Automated)	☐	☐
5.4	Ensure routing tables for VPC peering are "least access" (Manual)	☐	☐

Appendix: Change History

Date	Version	Changes for this version
2/1/2016	1.0.0	Initial Release
11/9/2016	1.1.0	Added recommendation 1.3 (Ticket #69)
11/9/2016	1.1.0	Updated section 3 remediations (Ticket #88)
11/9/2016	1.1.0	1.4 - Updated commands (Ticket #59)
11/9/2016	1.1.0	1.14 - Updated commands (Ticket #61)
11/9/2016	1.1.0	1.3 - Updated commands (Ticket #58)
11/9/2016	1.1.0	1.13 - Updated commands (Ticket #60)
11/9/2016	1.1.0	4.4 - Updated Remediation (Ticket #67, #68)
11/9/2016	1.1.0	2.1 - Updated remediation (Ticket #66)
11/9/2016	1.1.0	1.14 - Updated Title (Ticket #73)
11/9/2016	1.1.0	3.15 - Moved to section 1 (Ticket #93)
11/9/2016	1.1.0	4.3 - Added note (Ticket #106)
11/9/2016	1.1.0	2.7 - Fixed typo in description (Ticket #54)
11/9/2016	1.1.0	3.1 - Added note to remediation (Ticket #109)
11/9/2016	1.1.0	Added Recommendation 1.24 (Ticket 147 | P a g e #103)
11/9/2016	1.1.0	Standardized benchmark example variables (Ticket #65)
11/9/2016	1.1.0	1.14 - Updated audit (Ticket #76)
11/9/2016	1.1.0	4.1, 4.2, 4.5 - Added warning (Ticket #87)
11/9/2016	1.1.0	3.2 - Updated audit (Ticket #75, #108)
11/9/2016	1.1.0	1.11 - Updated remediation (Ticket #53)

11/9/2016	1.1.0	2.2 - Fixed typo in description (Ticket #57)
11/9/2016	1.1.0	1.1 - Fixed typo in description and rationale (Ticket #56)
11/9/2016	1.1.0	2.1, 2.5, 2.6, 4.3 - Added notes (Ticket #89)
5/23/2018	1.2.0	UPDATE - 2.2 - Ensure CloudTrail log file validation is enabled-ticket 6199
5/23/2018	1.2.0	UPDATE - 3.1 - Update/Correct Audit & Remediation: Filter Pattern - Remove HTML Literals-ticket 6227
5/23/2018	1.2.0	UPDATE - 3.3 - Update/correct Audit and Remediation by removing HTML literals from filters-ticket 6229
5/23/2018	1.2.0	UPDATE - 3.6 - Remove HTML literals/escape characters from filter pattern-ticket 6230
5/23/2018	1.2.0	UPDATE - 2.3 - Recommendation Title need to be corrected/Updated-ticket 6210
5/23/2018	1.2.0	UPDATE - 1.21 - Ensure IAM instance roles are used for AWS resource access from instances - Grammatical-ticket 2279
5/23/2018	1.2.0	UPDATE - Overview for section 3 Monitoring-ticket 6219
5/23/2018	1.2.0	UPDATE - 1.3 - Ensure credentials unused for 90 days or greater are disabled - Audit-ticket 6188
5/23/2018	1.2.0	UPDATE - 1.4 Ensure access keys are rotated every 90 days or less - Audit-ticket 6187
5/23/2018	1.2.0	UPDATE - Error in 3.14 Ensure a log metric filter and alarm exist for VPC changes-ticket 6070
5/23/2018	1.2.0	UPDATE - 3.7 Ensure a log metric filter and alarm exist for disabling.... filterPattern error-ticket 6008
5/23/2018	1.2.0	UPDATE - Audit Procedure for Effectiveness: 3.1 to 3.14-ticket 6212
5/23/2018	1.2.0	UPDATE - 2.1 - Multiregion Cloudtrail: Management Events-ticket 6217
5/23/2018	1.2.0	UPDATE - 1.20 - Set to "Not Scored" vs "Scored"-ticket 6166

5/23/2018	1.2.0	UPDATE - 1.18 - Set to "Not Scored" vs "Scored"-ticket 6164
5/23/2018	1.2.0	UPDATE - 2.1 - CLI commands are incomplete-ticket 4843
5/23/2018	1.2.0	UPDATE - 1.19 - Set to "Not Scored" vs "Scored"-ticket 6165
5/23/2018	1.2.0	UPDATE - 2.6 - Adding step in CLI audit just for the sake of completeness-ticket 6280
5/23/2018	1.2.0	UPDATE - 2.5 Need more clarification on CLI audit step 2-ticket 6279
5/23/2018	1.2.0	UPDATE - 2.5 - No references provided/updated audit-ticket4860
5/23/2018	1.2.0	UPDATE - 1.3 - Ensure credentials unused for 90 days or greater are disabled-ticket 6208
5/23/2018	1.2.0	MOVE - 4.3 - Ensure VPC flow logging is enabled in all VPCs" to section 2 "Logging"-ticket 2274
5/23/2018	1.2.0	DELETE - 3.15 - Ensure appropriate subscribers to each SNS topic -Not really a config item-ticket 4844
5/23/2018	1.2.0	DELETE - 1.17 - Enable detailed billing-ticket6336
5/23/2018	1.2.0	DELETE - 1.18 - Ensure IAM Master and IAM Manager roles are active-ticket 6371
5/23/2018	1.2.0	UPDATE - 2.2 - Ensure CloudTrail log file validation is enabled - Audit section-ticket 6200
5/23/2018	1.2.0	UPDATE - 1.18 - HTML embedded in the remediation procedure-ticket 6163
5/23/2018	1.2.0	UPDATE - 1.22 - Ensure IAM policies that allow... - False Positives ::UPDATE Audit, Use of == instead of contains-ticket 6350
5/23/2018	1.2.0	UPDATE - 2.3 - Ensure the S3 bucket used... - Principle set to *-ticket 6390
5/23/2018	1.2.0	UPDATE - Map CIS Controls Version 7 to all recommendations-ticket 6394
8-Jan-19	1.3.0	UPDATE-Avoid the use of the "root user" account-Additional reference added (Ticket 7157)

1-Apr-20	1.3.0	UPDATE - Do not setup access keys during initial user setup for all IAM users that have a console password - rule name is differed from audit procedure (Ticket 6837)
6-Apr-20	1.3.0	UPDATE - Ensure MFA is enabled for the "root user " account - add reference (Ticket 10147)
6-Apr-20	1.3.0	UPDATE - Ensure multi-factor authentication (MFA) is enabled for all IAM users that have a console password - add reference (Ticket 10136)
17-Apr-20	1.3.0	UPDATE - Ensure routing tables for VPC peering are "least access" - add reference (Ticket 10197)
28-Apr-20	1.3.0	UPDATE - Multiple Recommendations - GovCloud (US) regions do not have traditional 'root' account (Ticket 6490)
21-May-20	1.3.0	UPDATE - Ensure IAM password policy expires passwords within 90 days or less - Add reference (Ticket 10146)
21-May-20	1.3.0	UPDATE - Ensure IAM password policy prevents password reuse - Add reference (Ticket 10145)
21-May-20	1.3.0	UPDATE - Ensure IAM password policy requires minimum length of 14 or greater - Add reference (Ticket 10144)
28-May-20	1.3.0	UPDATE - Ensure a log metric filter and alarm exist for security group changes - Use of quotes instead of backticks in CLI commands (Ticket 8409)
28-May-20	1.3.0	UPDATE - Ensure no security groups allow ingress from 0.0.0.0/0 to port 22 - Add reference (Ticket 10193)
28-May-20	1.3.0	UPDATE - Ensure no security groups allow ingress from 0.0.0.0/0 to port 3389 - Add reference (Ticket 10194)
28-May-20	1.3.0	UPDATE - Ensure the default security group of every VPC restricts all traffic - Add reference (Ticket 10196)
2-Jun-20	1.3.0	UPDATE - Ensure that IAM Access analyzer is enabled - Cli to audit and/or remediation sections (Ticket 10767)
3-Jun-20	1.3.0	UPDATE - Ensure IAM policies are attached only to groups or roles - add auidt to include inline policies (Ticket 10892)

5-Jun-20	1.3.0	UPDATE - Ensure IAM policies that allow full "*:*" administrative privileges are not created - change title and audit to address attached policies (Ticket 8365)
8-Jun-20	1.3.0	DELETE - Ensure IAM password policy expires passwords within 90 days or less (Ticket 10883)
11-Jun-20	1.3.0	UPDATE - Reordering of IAM Section (Section 1) (Ticket 10612)
17-Jun-20	1.3.0	ADD - Ensure a log metric filter and alarm exists for AWS Organizations changes (Ticket 10894)
17-Jun-20	1.3.0	ADD - Ensure that IAM Access analyzer is enabled (Ticket 9671)
17-Jun-20	1.3.0	DELETE - Ensure IAM password policy require at least one lowercase letter (Ticket 10880)
17-Jun-20	1.3.0	DELETE - Ensure IAM password policy require at least one number (Ticket 10882)
17-Jun-20	1.3.0	DELETE - Ensure IAM password policy require at least one symbol (Ticket 10881)
17-Jun-20	1.3.0	DELETE - Ensure IAM password policy requires at least one uppercase letter (Ticket 10879)
24-Jun-20	1.3.0	ADD - Ensure that all the expired SSL/TLS certificates stored in AWS IAM are removed (Ticket 6936)
24-Jun-20	1.3.0	UPDATE - Ensure no root user account access key exists - Add new audit procedure (Ticket 10601)
24-Jun-20	1.3.0	Update - Various recommendations - all http links to https. (Ticket 7251)
7-Jul-20	1.3.0	UPDATE - Do not setup access keys during initial user setup for all IAM users that have a console password - Update description (Ticket 7108)
7-Jul-20	1.3.0	UPDATE - Ensure a log metric filter and alarm exist for Management Console sign-in without MFA - Update the filter pattern to minimize noise (Ticket 6742)

7-Jul-20	1.3.0	UPDATE - Ensure a support role has been created to manage incidents with AWS Support - GovCloud (US) regions do not have the "AWSSupportAccess" policy (Ticket 6491)
7-Jul-20	1.3.0	UPDATE - Ensure access keys are rotated every 90 days or less - Add audit procedure, reference (Ticket 10138)
7-Jul-20	1.3.0	UPDATE - Ensure credentials unused for 90 days or greater are disabled - add audit Procedure, Remediation Procedure, reference (Ticket 10137)
8-Jul-20	1.3.0	ADD - Ensure no Network ACLs allow ingress from 0.0.0.0/0 to port 22 (Ticket 10198)
8-Jul-20	1.3.0	ADD - Ensure no Network ACLs allow ingress from 0.0.0.0/0 to port 3389 (Ticket 10199)
8-Jul-20	1.3.0	ADD - Ensure that Object-level logging for read events is enabled for S3 bucket (Ticket 10704)
8-Jul-20	1.3.0	ADD - Ensure that Object-level logging for write events is enabled for S3 bucket (Ticket 10703)
8-Jul-20	1.3.0	UPDATE - Ensure S3 bucket access logging is enabled on the CloudTrail S3 bucket - recommend changing to CloudTrail object-level logging (Ticket 8398)
9-Jul-20	1.3.0	ADD - Ensure there is a maximum of only one active access key available for any single IAM user (Ticket 10705)
9-Jul-20	1.3.0	UPDATE - Ensure routing tables for VPC peering are "least access" - add AWS Transit Gateway routing guidance (Ticket 10927)
14-Jul-20	1.3.0	ADD - Ensure that S3 Buckets are configured with 'Block public access (bucket settings)' (Ticket 10921)
14-Jul-20	1.3.0	UPDATE - Eliminate use of the root user for administrative and daily tasks - Rewrite audit/change scoring status (Ticket 11070)
20-Jul-20	1.3.0	ADD - Simple Storage Service (S3) and Elastic Compute Cloud (EC2) sections (Ticket 11148)

20-Jul-20	1.3.0	UPDATE - Ensure a log metric filter and alarm exist for Management... - Additional granularity in control 3.2 for SAML authenticated users (Ticket 7726)
20-Jul-20	1.3.0	UPDATE - Ensure access keys are rotated every 90 days or less - change for Audit Procedure (Ticket 10939)
20-Jul-20	1.3.0	UPDATE - Ensure routing tables for VPC peering are "least access" - Reword Transit Gateway note (Ticket 10990)
20-Jul-20	1.3.0	UPDATE - Multiple recommendations in Networking section (Ticket 11094)
22-Jul-20	1.3.0	ADD - Ensure IAM users are managed centrally via identity federation or AWS Organizations for multi-account environments (Ticket 11173)
22-Jul-20	1.3.0	ADD - Ensure S3 Bucket Policy allows HTTPS requests (Ticket 7164)
22-Jul-20	1.3.0	ADD - Section 2 'Storage' (Ticket 11167)
22-Jul-20	1.3.0	UPDATE - Ensure a log metric filter and alarm exist for unauthorized API calls - Update metric filter to exclude HeadBucket event (Ticket 11084)