CENTER FOR INTERNET SECURITY

CIS Amazon Web Services Foundations

v1.0.0 - 02-29-2016

http://benchmarks.cisecurity.org

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 feedback@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://community.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

Scoring Information

A scoring status indicates whether compliance with the given recommendation impacts the assessed target's benchmark score. The following scoring statuses are used in this benchmark:

Scored
Failure to comply with "Scored" recommendations will decrease the final benchmark score. Compliance with "Scored" recommendations will increase the final benchmark score.

Not Scored
Failure to comply with "Not Scored" recommendations will not decrease the final benchmark score. Compliance with "Not Scored" recommendations will not increase the final benchmark score.

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:

Contributors
Adam Montville, *Center for Internet Security*
Cindy Spiess, *Adobe*
Blake Frantz
Gavin Fitzpatrick
Iben Rodriguez
Amol Pathak Senior Cyber Security Advisor
James Covington
Chris Launey
Rob Witoff, *Coinbase*
John Martinez, *Evident.io*
Tim Sandage
Mike de Libero, *MDE Development, Inc.*
Alex Corley

Recommendations

1 Identity and Access Management

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

1.1 Avoid the use of the "root" account (Scored)

Profile Applicability:
- Level 1

Description:
The "root" account is has unrestricted access to all resources in the AWS account. It is highly recommended that the use of this account be avoided.

Rationale:
The "root" account is the most privileged AWS account. Minimizing the use of this account and adopting the principle of least privilege for access management will reduce the risk of accidental changes and unintended disclosure of highly privileged credentials.

Audit:
Implement the `Ensure a log metric filter and alarm exist for usage of "root" account` recommendation in the `Monitoring` section of this benchmark to receive notifications of root account usage. Additionally, executing the following command will provide ad-hoc means for determining the last time the root account was used:

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

Note: there are a few conditions under which the use of the root account is required, such as requesting a penetration test or creating a CloudFront private key.

Remediation:
Follow the remediation instructions of the `Ensure IAM policies are attached only to groups or roles` recommendation.

References:

1. http://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html
2. CIS CSC v6.0 #5.1

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

Profile Applicability:
- Level 1

Description:
Multi-Factor Authentication (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. 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 emits 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:

Via Management 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 each user having a checkmark in the `Password` column also has a value in the `MFA Device` column.

Via the CLI

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 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:

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 A virtual MFA device, and then choose Next Step.
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 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. http://tools.ietf.org/html/rfc6238
2. http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa.html
3. CIS CSC v6.0 #5.6, #11.4, #12.6, #16.11

1.3 Ensure credentials unused for 90 days or greater are disabled (Scored)

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 removed or deactivated.

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:

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 users within an AWS Account - open this file
6. For each user having `password_enabled` set to `TRUE`, ensure `password_last_used` is less than `90` days ago.
7. For each user having `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.

Via the CLI

1. Run the following commands:

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

2. For each user having `password_enabled` set to `TRUE`,
   ensure `password_last_used_date` 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.

Remediation:
Perform the following to remove or deactivate credentials:

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 `Make Inactive` for credentials that have not been used in `90` Days
7. As an IAM User
    - Click on `Make Inactive` or `Delete` for credentials which have not been used in `90` Days

References:

1. CCE-78900-8
2. CIS CSC v6.0 #16.6

1.4 Ensure access keys are rotated every 90 days or less (Scored)

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:

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 Access Key usage for all IAM users within an AWS Account - open this file
6. Focus on the following columns (where x = 1 or 2)
    - `access_key_X_active`
    - `access_key_X_last_rotated`
    - `access_key_X_last_used_date`
7. Ensure all active keys have been rotated within `90` days
8. Ensure all active keys have been used since last rotation
    - Keys not in-use since last rotation should be disabled/deleted

Via the CLI

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

Remediation:
Perform the following to rotate 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. As an Administrator
    - Click on `Make Inactive` for keys that have not been rotated in `90` Days
7. As an IAM User
    - Click on `Make Inactive` or `Delete` for keys which have not been rotated or used in `90` Days
8. Click on `Create Access Key`
9. Update programmatic call with new Access Key credentials

Via the CLI

```
aws iam update-access-key
aws iam create-access-key
aws iam delete-access-key
```

References:

1. CCE-78902-4

1.5 Ensure IAM password policy requires at least one uppercase letter (Scored)

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 comprised of different character sets. It is recommended that the password policy require at least one uppercase letter.

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:

Via AWS 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 "Requires at least one uppercase letter" is checked under "Password Policy"

Via the CLI

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

Ensure the output of the above command includes "RequireUppercaseCharacters": true

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

Via AWS 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 "Requires at least one uppercase letter"
5. Click "Apply password policy"

References:
1. CCE-78903-2

1.6 Ensure IAM password policy require at least one lowercase letter (Scored)

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 comprised of different character sets. It is recommended that the password policy require at least one lowercase letter.

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:

Via the AWS 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 "Requires at least one lowercase letter" is checked under "Password Policy"

Via the CLI

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

Ensure the output of the above command includes "RequireLowercaseCharacters": true

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

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 "Requires at least one lowercase letter"
5. Click "Apply password policy"

References:

1. CCE-78904-0

1.7 Ensure IAM password policy require at least one symbol (Scored)

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 comprised of different character sets. It is recommended that the password policy require at least one symbol.

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:

Via AWS 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 "Require at least one non-alphanumeric character" is checked under "Password Policy"

Via the CLI

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

Ensure the output of the above command includes "RequireSymbols": true

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

Via AWS 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 "Require at least one non-alphanumeric character"
5. Click "Apply password policy"

References:

1. CCE-78905-7

1.8 Ensure IAM password policy require at least one number (Scored)

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 comprised of different character sets. It is recommended that the password policy require at least one number.

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:

Via AWS 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 "Require at least one number " is checked under "Password Policy"

Via the CLI

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

Ensure the output of the above command includes "RequireNumbers": true

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

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 "Require at least one number"
5. Click "Apply password policy"

References:

1. CCE-78906-5

1.9 Ensure IAM password policy requires minimum length of 14 or greater (Scored)

Profile Applicability:
- Level 2

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:

Via AWS 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.

Via the CLI

```
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:

Via AWS 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"

References:

1. CCE-78907-3
2. CIS CSC v6.0 #5.7, #16.12

1.10 Ensure IAM password policy prevents password reuse (Scored)

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:

Via AWS 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

Via the CLI

```
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:

Via AWS 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`

References:

1. CCE-78908-1

1.11 Ensure IAM password policy expires passwords within 90 days or less (Scored)

Profile Applicability:
- Level 1

Description:
IAM password policies can require passwords to be rotated or expired after a given number of days. It is recommended that the password policy expire passwords after 90 days or less.

Rationale:
Reducing the password lifetime increases account resiliency against brute force login attempts. Additionally, requiring regular password changes help in the following scenarios:

- Passwords can be stolen or compromised sometimes without your knowledge. This can happen via a system compromise, software vulnerability, or internal threat.
- Certain corporate and government web filters or proxy servers have the ability to intercept and record traffic even if it's encrypted.
- Many people use the same password for many systems such as work, email, and personal.
- Compromised end user workstations might have a keystroke logger.

Audit:
Perform the following to ensure the password policy is configured as prescribed:

Via AWS 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 "Enable password expiration" is checked
5. Ensure "Password expiration period (in days):" is set to 90 or less

Via the CLI

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

Ensure the output of the above command includes "MaxPasswordAge": 90 or less

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

Via AWS 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 "Enable password expiration”
5. Set "Password expiration period (in days):" is set to `90`

References:

1. CCE-78909-9

1.12 Ensure no root account access key exists (Scored)

Profile Applicability:
- Level 1

Description:
The root 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 account be removed.

Rationale:
Removing access keys associated with the root 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 account has access keys:

Via the AWS 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`.

Via the CLI

1. Run the following commands:

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

2. For the `<root_account>` user, ensure the `access_key_1_active` and `access_key_2_active` fields are set to `FALSE`.

Remediation:

Perform the following to delete or disable active root access keys being

Via the AWS 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 *Account Name* <*value*> at the top left and select 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. CIS CSC v6.0 #5.1

1.13 Ensure hardware MFA is enabled for the "root" account (Scored)

Profile Applicability:
- Level 2

Description:
The root 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. It is recommended that the root 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.

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

1. Run the following command:

```
aws iam get-account-summary
```

2. Ensure the AccountMFAEnabled property is set to 1

Remediation:
Perform the following to establish a hardware MFA for the root 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 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.

References:

1. CCE-78911-5
2. CIS CSC v6.0 #5.6, #11.4, #12.6, #16.11

1.14 Ensure security questions are registered in the AWS account (Not Scored)

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" 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 login access.

Audit:
Perform the following in the AWS Management Console:

1. Login to the AWS account as root
2. On the top right you will see the <Account Name>
3. Click on the <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:
Perform the following in the AWS Management Console:

1. Login to the AWS Account as root
2. Select the 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

1.15 Ensure IAM policies are attached only to groups or roles (Scored)

Profile Applicability:
- Level 1

Description:
By default, IAM users, groups, and roles have no access to AWS resources. IAM policies are the means by which privileges are granted to users, groups, or roles. It is recommended that IAM policies be applied directly to groups and roles but not users.

Rationale:
Assigning privileges at the group or role level reduces the complexity of access management as the number of users grow. Reducing access management complexity may in-turn reduce opportunity for a principal to inadvertently receive or retain excessive privileges.

Audit:
Perform the following to determine if policies are attached directly 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  <username>
aws iam list-user-policies --user-name <username>
```

3. If any policies are returned, the user has a 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:
    1. Select the user
    2. Click on the `Permissions` tab
    3. Expand `Managed Policies`
    4. Click `Detach Policy` for each policy
    5. Expand `Inline Policies`
    6. Click `Remove Policy` for each policy

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

2 Logging

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

2.1 Ensure CloudTrail is enabled in all regions (Scored)

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.

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

Via the management 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
    1. 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`

Via the CLI

```
aws cloudtrail describe-trails
```

Ensure `IsMultiRegionTrail` is set to `true`

Remediation:
Perform the following to enable global CloudTrail logging:

Via the management 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
    1. Click the edit icon (pencil) next to `Apply trail to all regions`
    2. Click `Yes`
    3. Click `Save`

Via the CLI

```
aws cloudtrail create-trail --name <s3-bucket-name> --is-multi-region-trail
aws cloudtrail update-trail --name --is-multi-region-trail
```

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

Default Value:
Not Enabled

References:

1. CCE-78913-1
2. CIS CSC v6.0 #14.6

2.2 Ensure CloudTrail log file validation is enabled (Scored)

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:

Via the management 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
    1. 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. Under the `S3` section, ensure `Enable log file validation` is set to `Yes`

Via the CLI

```
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:

Via the management 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`

Via the CLI

```
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 <arn> --start-time <time> --end-time <time>
```

Default Value:
Not Enabled

References:

1. http://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-log-file-validation-enabling.html
2. CCE-78914-9
3. CIS CSC v6.0 #6.3

2.3 Ensure the S3 bucket CloudTrail logs to is not publicly accessible (Scored)

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 prevents 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:

Via the Management 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 `*`.

Via the CLI:

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 <bucket_name_from_step_1> --query
'Grants[?Grantee.URI==`http://acs.amazonaws.com/groups/global/AllUsers`]'
```

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

```
aws s3api get-bucket-acl --bucket <bucket_name_from_step_1> --query
'Grants[?Grantee.URI==`http://acs.amazonaws.com/groups/global/Authenticated
Users`]'
```

4. Get the S3 Bucket Policy

```
aws s3api get-bucket-policy --bucket <bucket_name_from_step_1>
```

5. Ensure the policy does not contain a `Statement` having an `Effect` set to `Allow` and a `Principal` set to `*`.

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 `*`.

Default Value:
By default, S3 buckets are not publicly accessible

References:

1. CCE-78915-6

2.4 Ensure CloudTrail trails are integrated with CloudWatch Logs (Scored)

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.

Audit:
Perform the following to ensure CloudTrail is configured as prescribed:

Via the AWS management 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.

Via the CLI

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 <name>
```

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

Remediation:
Perform the following to establish the prescribed state:

Via the AWS management 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
    1. Create/Select an `IAM Role` and `Policy Name`
    2. Click `Allow` to continue

Via the CLI

```
aws cloudtrail update-trail --name <name> --cloudwatch-logs-log-group-arn <group_arn> --
cloudwatch-logs-role-arn <role_arn>
```

Impact:
CloudWatch Logs will be stored indefinitely unless a specific retention period is defined by the user. See the following AWS resource to manage CloudWatch Logs retention periods:

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

References:
1. https://aws.amazon.com/cloudtrail/
2. CCE-78916-4
3. CIS CSC v6.0 #6.6, #14.6

2.5 Ensure AWS Config is enabled in all regions (Scored)

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

Via AWS Management 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.

Remediation:
Perform the following in the AWS Management 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

API Call:

```
aws configservice start-configuration-recorder
```

References:

1. CCE-78917-2
2. CIS CSC v6.0 #1.1, #1.3, #1.4, #5.2, #11.1 - #11.3, #14.6

2.6 Ensure S3 bucket access logging is enabled on the CloudTrail S3 bucket (Scored)

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 an 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:

Via the management 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.

Via the CLI

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

Remediation:
Perform the following to enable S3 bucket logging:

Via the Management 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:<bucket_name>` click on `Logging`
5. Configure bucket logging
    1. Click on `Enabled` checkbox
    2. Select Target Bucket from list
    3. Enter a Target Prefix
6. Click `Save`

Default Value:
Logging is disabled.

References:

1. CCE-78918-0
2. CIS CSC v6.0 #14.6

2.7 Ensure CloudTrail logs are encrypted at rest using KMS CMKs (Scored)

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.

Audit:

Perform the following to determine if CloudTrail is configured to use SSE-KMS:

Via the Management 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.

Via the CLI

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:

Via the Management 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`

Via the CLI

```
aws cloudtrail update-trail --name <value>  --kms-id <value>
aws kms put-key-policy --key-id <id> --policy <value>
```

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

References:

1. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/encrypting-cloudtrail-log-files-with-aws-kms.html
2. http://docs.aws.amazon.com/kms/latest/developerguide/create-keys.html
3. CCE-78919-8
4. CIS CSC v6.0 #14.5

2.8 Ensure rotation for customer created CMKs is enabled (Scored)

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:
Via the Management 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.

Via the CLI

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 <key_id>
```

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

Remediation:
Via the Management 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.

Via the CLI

1. Run the following command to enable key rotation:

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

References:

1. https://aws.amazon.com/kms/pricing/
2. http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf
3. CCE-78920-6

3 Monitoring

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 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.

3.1 Ensure a log metric filter and alarm exist for unauthorized API calls (Scored)

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.

Audit:
Perform the following to determine if the account is configured as prescribed:

1. Identify the log group name configured for use with CloudTrail:

```
aws cloudtrail describe-trails
```

2. Note the *<group>* value associated with `CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:1234567890:log-group:,<group>:*"
```

3. Get a list of all associated metric filters for this *<group>*:

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

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

```
"filterPattern": "{ ($.errorCode = \"*UnauthorizedOperation\") || ($.errorCode =
\"AccessDenied*\") }"
```

5. Note the `metricName` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `metricName` captured in step 4.

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

7. Note the `AlarmActions` value - this will provide the SNS topic ARN value.

8. Ensure there is at least one subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <value-from-step-7>
```

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 *<group>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <group> --filter-name <name> --metric-
transformations <value> --filter-pattern '{ ($.errorCode = "*UnauthorizedOperation")
|| ($.errorCode = "AccessDenied*") }'
```

2. Create an SNS topic that the alarm will notify

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

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

```
aws sns subscribe --topic-arn arn:aws:sns:us-west-2:<account_id>:<topic_name> --
protocol email --notification-endpoint <my-email@example.com>
```

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 <value> --metric-name <name_from_step_1>
--statistic Sum --period 300 --threshold 1 --comparison-operator
GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace <CloudTrailMetrics> -
-alarm-actions <topic_arn_from_step_3>
```

References:

1. https://aws.amazon.com/sns/
2. CCE-79186-3

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

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 determine if the account is configured as prescribed:

1. Identify the log group name configured for use with CloudTrail:

```
aws cloudtrail describe-trails
```

2. Note the *<group>* value associated with `CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:1234567890:log-group:,<group>:*",
```

3. Get a list of all associated metric filters for this *<group>*:

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

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

```
"filterPattern": "{ $.userIdentity.sessionContext.attributes.mfaAuthenticated !=
"true" }"
```

5. Note the `metricName` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `metricName` captured in step 4.

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

7. Note the `AlarmActions` value - this will provide the SNS topic ARN value.

8. Ensure there is at least one subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <value-from-step-7>
```

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 *<group>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <group> --filter-name <name> --metric-
transformations <value> --filter-pattern '{
$.userIdentity.sessionContext.attributes.mfaAuthenticated != "true" }'
```

2. Create an SNS topic that the alarm will notify

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

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

```
aws sns subscribe --topic-arn arn:aws:sns:us-west-2:<account_id>:<topic_name> --
protocol email --notification-endpoint <my-email@example.com>
```

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 <value> --metric-name <name_from_step_1>
--statistic Sum --period 300 --threshold 1 --comparison-operator
GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace <CloudTrailMetrics> -
-alarm-actions <topic_arn_from_step_3>
```

References:

1. http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/viewing_metrics_with_cloudwatch.html
2. CCE-79187-1
3. CIS CSC v6.0 #5.5

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

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 determine if the account is configured as prescribed:

1. Identify the log group name configured for use with CloudTrail:

```
aws cloudtrail describe-trails
```

2. Note the *<group>* value associated with `CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<account_number>:log-group:,<group>:*",
```

3. Get a list of all associated metric filters for this *<group>*:

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

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

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

5. Note the `metricName` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `metricName` captured in step 4.

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

7. Note the `AlarmActions` value - this will provide the SNS topic ARN value.

8. Ensure there is at least one subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <value-from-step-7>
```

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 *<group>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <group> --filter-name <name> --metric-
transformations <value> --filter-pattern '{ $.userIdentity.type = "Root" &&
$.userIdentity.invokedBy NOT EXISTS && $.eventType != "AwsServiceEvent" } '
```

2. Create an SNS topic that the alarm will notify

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

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

```
aws sns subscribe --topic-arn arn:aws:sns:us-west-2:<account_id>:<topic_name> --
protocol email --notification-endpoint <my-email@example.com>
```

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 <value> --metric-name <name_from_step_1>
--statistic Sum --period 300 --threshold 1 --comparison-operator
GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace <CloudTrailMetrics> -
-alarm-actions <topic_arn_from_step_3>
```

References:

1. CCE-79188-9
2. CIS CSC v6.0 #4.6, #5.1, #5.5

3.4 Ensure a log metric filter and alarm exist for IAM policy changes (Scored)

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 determine if the account is configured as prescribed:

1. Identify the log group name configured for use with CloudTrail:

```
aws cloudtrail describe-trails
```

2. Note the *<group>* value associated with `CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<account_number>:log-group:,<group>:*",
```

3. Get a list of all associated metric filters for this *<group>*:

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

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

```
"filterPattern":
"{($.eventName=DeleteGroupPolicy)||($.eventName=DeleteRolePolicy)||($.eventName=Delete
UserPolicy)||($.eventName=PutGroupPolicy)||($.eventName=PutRolePolicy)||($.eventName=P
utUserPolicy)||($.eventName=CreatePolicy)||($.eventName=DeletePolicy)||($.eventName=Cr
eatePolicyVersion)||($.eventName=DeletePolicyVersion)||($.eventName=AttachRolePolicy)|
|($.eventName=DetachRolePolicy)||($.eventName=AttachUserPolicy)||($.eventName=DetachUs
erPolicy)||($.eventName=AttachGroupPolicy)||($.eventName=DetachGroupPolicy)}"
```

5. Note the `metricName` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `metricName` captured in step 4.

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

7. Note the `AlarmActions` value - this will provide the SNS topic ARN value.

8. Ensure there is at least one subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <value-from-step-7>
```

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 *<group>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <group> --filter-name <name> --metric-
transformations <value> --filter-pattern
'{($.eventName=DeleteGroupPolicy)||($.eventName=DeleteRolePolicy)||($.eventName=Delete
UserPolicy)||($.eventName=PutGroupPolicy)||($.eventName=PutRolePolicy)||($.eventName=P
utUserPolicy)||($.eventName=CreatePolicy)||($.eventName=DeletePolicy)||($.eventName=Cr
eatePolicyVersion)||($.eventName=DeletePolicyVersion)||($.eventName=AttachRolePolicy)|
|($.eventName=DetachRolePolicy)||($.eventName=AttachUserPolicy)||($.eventName=DetachUs
erPolicy)||($.eventName=AttachGroupPolicy)||($.eventName=DetachGroupPolicy)}'
```

2. Create an SNS topic that the alarm will notify

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

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

```
aws sns subscribe --topic-arn arn:aws:sns:us-west-2:<account_id>:<topic_name> --
protocol email --notification-endpoint <my-email@example.com>
```

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 <value> --metric-name <name_from_step_1>
--statistic Sum --period 300 --threshold 1 --comparison-operator
GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace <CloudTrailMetrics> -
-alarm-actions <topic_arn_from_step_3>
```

References:

1. CCE-79189-7

3.5 Ensure a log metric filter and alarm exist for CloudTrail configuration changes (Scored)

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 determine if the account is configured as prescribed:

1. Identify the log group name configured for use with CloudTrail:

```
aws cloudtrail describe-trails
```

2. Note the <*group*> value associated with `CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<account_number>:log-group:,<group>:*",
```

3. Get a list of all associated metric filters for this <*group*>:

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

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

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

5. Note the `metricName` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `metricName` captured in step 4.

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

7. Note the `AlarmActions` value - this will provide the SNS topic ARN value.

8. Ensure there is at least one subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <value-from-step-7>
```

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 <*group*> taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <group> --filter-name <name> --metric-
transformations <value> --filter-pattern '{ ($.eventName = CreateTrail) ||
($.eventName = UpdateTrail) || ($.eventName = DeleteTrail) || ($.eventName =
StartLogging) || ($.eventName = StopLogging) } '
```

2. Create an SNS topic that the alarm will notify

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

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

```
aws sns subscribe --topic-arn arn:aws:sns:us-west-2:<account_id>:<topic_name> --
protocol email --notification-endpoint <my-email@example.com>
```

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 <value> --metric-name <name_from_step_1>
--statistic Sum --period 300 --threshold 1 --comparison-operator
GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace <CloudTrailMetrics> -
-alarm-actions <topic_arn_from_step_3>
```

References:

1. CCE-79190-5

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

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 determine if the account is configured as prescribed:

1. Identify the log group name configured for use with CloudTrail:

```
aws cloudtrail describe-trails
```

2. Note the <*group*> value associated with `CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<account_number>:log-group:,<group>:*",
```

3. Get a list of all associated metric filters for this <*group*>:

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

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

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

5. Note the `metricName` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `metricName` captured in step 4.

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

7. Note the `AlarmActions` value - this will provide the SNS topic ARN value.

8. Ensure there is at least one subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <value-from-step-7>
```

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 `<group>` taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <group> --filter-name <name> --metric-
transformations <value> --filter-pattern '{ ($.eventName = ConsoleLogin) &&
($.errorMessage = "Failed authentication") }'
```

2. Create an SNS topic that the alarm will notify

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

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

```
aws sns subscribe --topic-arn arn:aws:sns:us-west-2:<account_id>:<topic_name> --
protocol email --notification-endpoint <my-email@example.com>
```

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 <value> --metric-name <name_from_step_1>
--statistic Sum --period 300 --threshold 10 --comparison-operator
GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace <CloudTrailMetrics> -
-alarm-actions <topic_arn_from_step_3>
```

References:

1. CCE-79191-3

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

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 determine if the account is configured as prescribed:

1. Identify the log group name configured for use with CloudTrail:

```
aws cloudtrail describe-trails
```

2. Note the <*group*> value associated with `CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<account_number>:log-group:,<group>:*",
```

3. Get a list of all associated metric filters for this <*group*>:

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

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

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

5. Note the `metricName` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `metricName` captured in step 4.

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

7. Note the `AlarmActions` value - this will provide the SNS topic ARN value.

8. Ensure there is at least one subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <value-from-step-7>
```

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 <*group*> taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <group> --filter-name <name> --metric-
transformations <value> --filter-pattern '{($.eventSource = kms.amazonaws.com) &&
(($.eventName=DisableKey)||($.eventName=ScheduleKeyDeletion))}'
```

2. Create an SNS topic that the alarm will notify

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

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

```
aws sns subscribe --topic-arn arn:aws:sns:us-west-2:<account_id>:<topic_name> --
protocol email --notification-endpoint <my-email@example.com>
```

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 <value> --metric-name <name_from_step_1>
--statistic Sum --period 300 --threshold 1 --comparison-operator
GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace <CloudTrailMetrics> -
-alarm-actions <topic_arn_from_step_3>
```

References:

1. CCE-79192-1

3.8 Ensure a log metric filter and alarm exist for S3 bucket policy changes (Scored)

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 determine if the account is configured as prescribed:

1. Identify the log group name configured for use with CloudTrail:

```
aws cloudtrail describe-trails
```

2. Note the *<group>* value associated with `CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:1234567890:log-group:,<group>:*",
```

3. Get a list of all associated metric filters for this *<group>*:

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

4. 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)) }"
```

5. Note the `metricName` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `metricName` captured in step 4.

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

7. Note the `AlarmActions` value - this will provide the SNS topic ARN value.

8. Ensure there is at least one subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <value-from-step-7>
```

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 *<group>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <group> --filter-name <value> --metric-
transformations <value> --filter-pattern '{ ($.eventSource = s3.amazonaws.com) &&
(($.eventName = PutBucketAcl) || ($.eventName = PutBucketPolicy) || ($.eventName =
PutBucketCors) || ($.eventName = PutBucketLifecycle) || ($.eventName =
PutBucketReplication) || ($.eventName = DeleteBucketPolicy) || ($.eventName =
DeleteBucketCors) || ($.eventName = DeleteBucketLifecycle) || ($.eventName =
DeleteBucketReplication)) }'
```

2. Create an SNS topic

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

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

```
aws sns subscribe --topic-arn arn:aws:sns:us-west-2:<account_id>:<topic_name> --
protocol email --notification-endpoint <my-email@example.com>
```

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

```
aws cloudwatch put-metric-alarm --alarm-name <value> --metric-name <name_from_step_1>
--statistic Sum --period 300 --threshold 1 --comparison-operator
GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace <CloudTrailMetrics> -
-alarm-actions <topic_arn_from_step_3>
```

References:

1. CCE-79193-9

3.9 Ensure a log metric filter and alarm exist for AWS Config configuration changes (Scored)

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 determine if the account is configured as prescribed:

1. Identify the log group name configured for use with CloudTrail:

```
aws cloudtrail describe-trails
```

2. Note the *<group>* value associated with `CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<account_number>:log-group:,<group>:*",
```

3. Get a list of all associated metric filters for this *<group>*:

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

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

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

5. Note the `metricName` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `metricName` captured in step 4.

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

7. Note the `AlarmActions` value - this will provide the SNS topic ARN value.

8. Ensure there is at least one subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <value-from-step-7>
```

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 <*group*> taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <group> --filter-name <name> --metric-
transformations <value> --filter-pattern '{($.eventSource = config.amazonaws.com) &&
(($.eventName=StopConfigurationRecorder)||($.eventName=DeleteDeliveryChannel)||($.even
tName=PutDeliveryChannel)||($.eventName=PutConfigurationRecorder))}'
```

2. Create an SNS topic that the alarm will notify

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

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

```
aws sns subscribe --topic-arn arn:aws:sns:us-west-2:<account_id>:<topic_name> --
protocol email --notification-endpoint <my-email@example.com>
```

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 <value> --metric-name <name_from_step_1>
--statistic Sum --period 300 --threshold 1 --comparison-operator
GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace <CloudTrailMetrics> -
-alarm-actions <topic_arn_from_step_3>
```

References:

1. CCE-79194-7
2. CIS CSC v6.0 #5.4

3.10 Ensure a log metric filter and alarm exist for security group changes (Scored)

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 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 determine if the account is configured as prescribed:

1. Identify the log group name configured for use with CloudTrail:

```
aws cloudtrail describe-trails
```

2. Note the <*group*> value associated with `CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<account_number>:log-group:,<group>:*",
```

3. Get a list of all associated metric filters for this <*group*>:

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

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

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

5. Note the `metricName` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `metricName` captured in step 4.

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

7. Note the `AlarmActions` value - this will provide the SNS topic ARN value.

8. Ensure there is at least one subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <value-from-step-7>
```

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 Group changes and the *<group>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <group> --filter-name <name> --metric-
transformations <value> --filter-pattern '{ ($.eventName =
AuthorizeSecurityGroupIngress) || ($.eventName = AuthorizeSecurityGroupEgress) ||
($.eventName = RevokeSecurityGroupIngress) || ($.eventName =
RevokeSecurityGroupEgress) || ($.eventName = CreateSecurityGroup) || ($.eventName =
DeleteSecurityGroup)}'
```

2. Create an SNS topic that the alarm will notify

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

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

```
aws sns subscribe --topic-arn arn:aws:sns:us-west-2:<account_id>:<topic_name> --
protocol email --notification-endpoint <my-email@example.com>
```

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 <value> --metric-name <name_from_step_1>
--statistic Sum --period 300 --threshold 1 --comparison-operator
GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace <CloudTrailMetrics> -
-alarm-actions <topic_arn_from_step_3>
```

References:

1. CCE-79195-4

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

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 determine if the account is configured as prescribed:

1. Identify the log group name configured for use with CloudTrail:

```
aws cloudtrail describe-trails
```

2. Note the <*group*> value associated with `CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<account_number>:log-group:,<group>:*",
```

3. Get a list of all associated metric filters for this <*group*>:

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

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

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

5. Note the `metricName` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `metricName` captured in step 4.

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

7. Note the `AlarmActions` value - this will provide the SNS topic ARN value.

8. Ensure there is at least one subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <value-from-step-7>
```

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 <*group*> taken from audit step 2 above.

```
aws logs put-metric-filter --log-group-name <group> --filter-name <name> --metric-
transformations <value> --filter-pattern '{ ($.eventName = CreateNetworkAcl) ||
($.eventName = CreateNetworkAclEntry) || ($.eventName = DeleteNetworkAcl) ||
($.eventName = DeleteNetworkAclEntry) || ($.eventName = ReplaceNetworkAclEntry) ||
($.eventName = ReplaceNetworkAclAssociation) }'
```

2. Create an SNS topic that the alarm will notify

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

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

```
aws sns subscribe --topic-arn arn:aws:sns:us-west-2:<account_id>:<topic_name> --
protocol email --notification-endpoint <my-email@example.com>
```

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 <value> --metric-name <name_from_step_1>
--statistic Sum --period 300 --threshold 1 --comparison-operator
GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace <CloudTrailMetrics> -
-alarm-actions <topic_arn_from_step_3>
```

References:

1. CCE-79196-2

3.12 Ensure a log metric filter and alarm exist for changes to network gateways (Scored)

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 determine if the account is configured as prescribed:

1. Identify the log group name configured for use with CloudTrail:

```
aws cloudtrail describe-trails
```

2. Note the <*group*> value associated with `CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<account_number>:log-group:,<group>:*",
```

3. Get a list of all associated metric filters for this <*group*>:

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

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

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

5. Note the `metricName` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `metricName` captured in step 4.

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

7. Note the `AlarmActions` value - this will provide the SNS topic ARN value.

8. Ensure there is at least one subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <value-from-step-7>
```

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 Gateway changes and the *<group>* taken from audit step 2 above.

```
aws logs put-metric-filter --log-group-name <group> --filter-name <name> --metric-
transformations <value> --filter-pattern '{ ($.eventName = CreateCustomerGateway) ||
($.eventName = DeleteCustomerGateway) || ($.eventName = AttachInternetGateway) ||
($.eventName = CreateInternetGateway) || ($.eventName = DeleteInternetGateway) ||
($.eventName = DetachInternetGateway) }'
```

2. Create an SNS topic that the alarm will notify

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

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

```
aws sns subscribe --topic-arn arn:aws:sns:us-west-2:<account_id>:<topic_name> --
protocol email --notification-endpoint <my-email@example.com>
```

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 <value> --metric-name <name_from_step_1>
--statistic Sum --period 300 --threshold 1 --comparison-operator
GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace <CloudTrailMetrics> -
-alarm-actions <topic_arn_from_step_3>
```

References:

1. CCE-79197-0

3.13 Ensure a log metric filter and alarm exist for route table changes (Scored)

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 determine if the account is configured as prescribed:

1. Identify the log group name configured for use with CloudTrail:

```
aws cloudtrail describe-trails
```

2. Note the <*group*> value associated with `CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<account_number>:log-group:,<group>:*",
```

3. Get a list of all associated metric filters for this <*group*>:

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

4. 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) }"
```

5. Note the `metricName` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `metricName` captured in step 4.

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

7. Note the `AlarmActions` value - this will provide the SNS topic ARN value.

8. Ensure there is at least one subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <value-from-step-7>
```

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 <*group*> taken from audit step 2 above.

```
aws logs put-metric-filter --log-group-name <group> --filter-name <name> --metric-
transformations <value> --filter-pattern '{ ($.eventName = CreateRoute) ||
($.eventName = CreateRouteTable) || ($.eventName = ReplaceRoute) || ($.eventName =
ReplaceRouteTableAssociation) || ($.eventName = DeleteRouteTable) || ($.eventName =
DeleteRoute) || ($.eventName = DisassociateRouteTable) }'
```

2. Create an SNS topic that the alarm will notify

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

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

```
aws sns subscribe --topic-arn arn:aws:sns:us-west-2:<account_id>:<topic_name> --
protocol email --notification-endpoint <my-email@example.com>
```

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 <value> --metric-name <name_from_step_1>
--statistic Sum --period 300 --threshold 1 --comparison-operator
GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace <CloudTrailMetrics> -
-alarm-actions <topic_arn_from_step_3>
```

References:

1. CCE-79198-8

3.14 Ensure a log metric filter and alarm exist for VPC changes (Scored)

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

Audit:
Perform the following to determine if the account is configured as prescribed:

1. Identify the log group name configured for use with CloudTrail:

```
aws cloudtrail describe-trails
```

2. Note the *<group>* value associated with `CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<account_number>:log-group:,<group>:*",
```

3. Get a list of all associated metric filters for this *<group>*:

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

4. 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) }"
```

5. Note the `metricName` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `metricName` captured in step 4.

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

7. Note the `AlarmActions` value - this will provide the SNS topic ARN value.

8. Ensure there is at least one subscriber to the SNS topic

```
aws sns list-subscriptions-by-topic --topic-arn <value-from-step-7>
```

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 *<group>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <group> --filter-name <name> --metric-
transformations <value> --filter-pattern '{ ($.eventName = CreateVpc) || ($.eventName
= DeleteVpc) || ($.eventName = ModifyVpcAttribute) || ($.eventName =
AcceptVpcPeeringConnection) || ($.eventName = CreateVpcPeeringConnection) ||
($.eventName = DeleteVpcPeeringConnection) || ($.eventName =
RejectVpcPeeringConnection) || ($.eventName = AttachClassicLinkVpc) || ($.eventName =
DetachClassicLinkVpc) || ($.eventName = DisableVpcClassicLink) || ($.eventName =
EnableVpcClassicLink) }'
```

2. Create an SNS topic that the alarm will notify

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

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

```
aws sns subscribe --topic-arn arn:aws:sns:us-west-2:<account_id>:<topic_name> --
protocol email --notification-endpoint <my-email@example.com>
```

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 <value> --metric-name <name_from_step_1>
--statistic Sum --period 300 --threshold 1 --comparison-operator
GreaterThanOrEqualToThreshold --evaluation-periods 1 --namespace <CloudTrailMetrics> -
-alarm-actions <topic_arn_from_step_3>
```

References:

1. CCE-79199-6

3.15 Ensure security contact information is registered (Scored)

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 in the AWS Management Console to determine if security contact information is present:

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 in the AWS Management Console to establish security contact information:

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

3.16 Ensure appropriate subscribers to each SNS topic (Not Scored)

Profile Applicability:
- Level 1

Description:
AWS Simple Notification Service (SNS) is a web service that can publish messages from an application and immediately deliver them to subscribers or other applications. Subscribers are clients interested in receiving notifications from topics of interest; they can subscribe to a topic or be subscribed by the topic owner. When publishers have information or updates to notify their subscribers about, they can publish a message to the topic – which immediately triggers Amazon SNS to deliver the message to all applicable subscribers. It is recommended that the list of subscribers to given topics be periodically reviewed for appropriateness.

Rationale:
Reviewing subscriber topics will help ensure that only expected recipients receive information published to SNS topics.

Audit:
Perform the following to ensure appropriate subscribers:

Via the AWS Management console:

1. Sign in to the AWS Management Console and open the SNS console at https://console.aws.amazon.com/sns/
2. Click on `Topics` in the left navigation pane
3. Evaluate Topics by clicking on the value within the `ARN` column
    - Within a selected Topic evaluate:
        - Topic owner
        - Region
    - Within the `Subscriptions` section evaluate:
        - *Subscription ID*
        - *Protocol*
        - *Endpoint*
        - *Subscriber* (Account ID)

Via the CLI:

```
aws sns list-topics
aws sns list-subscriptions-by-topic --topic-arn <topic_arn>
```

Remediation:
Perform the following to remove undesired subscriptions:

Via Management Console

1. Sign in to the AWS Management Console and open the SNS console at https://console.aws.amazon.com/sns/
2. Click on `Subscriptions` in the left navigation pane
3. For any undesired subscription, select the corresponding checkboxes
4. Click `Actions`
5. Click `Delete Subscriptions`

Default Value:
Not Applicable

References:

1. https://aws.amazon.com/sns/

4 Networking

This section contains recommendations for configuring security-related aspects of the default Virtual Private Cloud (VPC)

4.1 Ensure no security groups allow ingress from 0.0.0.0/0 to port 22 (Scored)

Profile Applicability:
- Level 1

Description:
Security groups provide stateful filtering of ingress/egress network traffic to AWS resources. It is recommended that no security group allows unrestricted ingress access to port `22`.

Rationale:
Removing unfettered connectivity to remote console services, such as SSH, reduces a server's exposure to risk.

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:
    1. Select the security group
    2. Click the `Inbound Rules` tab
    3. Ensure no rule exists that has a port range that includes port `22` 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`.

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:
    1. Select the security group
    2. Click the `Inbound Rules` tab
    3. Identify the rules to be removed
    4. Click the `x` in the `Remove` column
    5. Click `Save`

4.2 Ensure no security groups allow ingress from 0.0.0.0/0 to port 3389 (Scored)

Profile Applicability:
- Level 1

Description:
Security groups provide stateful filtering of ingress/egress network traffic to AWS resources. It is recommended that no security group allows unrestricted ingress access to port `3389`.

Rationale:
Removing unfettered connectivity to remote console services, such as RDP, reduces a server's exposure to risk.

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:
    1. Select the security group
    2. Click the `Inbound Rules` tab
    3. Ensure no rule exists that has a port range that includes port `3389` and has a `Source` of `0.0.0.0/0`

Note: A `Port` value of `ALL` or a port range such as `1024-4098` are inclusive of port `3389`.

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:
    1. Select the security group
    2. Click the `Inbound Rules` tab
    3. Identify the rules to be removed
    4. Click the `x` in the `Remove` column
    5. Click `Save`

4.3 Ensure VPC Flow Logging is Enabled in all Applicable Regions (Scored)

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 the VPC.

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.

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

Via the Management 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:

Via the Management 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. Enter in a `Role` and `Destination Log Group`
8. Click `Create Log Flow`
9. Click on `CloudWatch Logs Group`

Note: By default, CloudWatch Logs will store VPC Flow Logs indefinitely unless a specific retention period is defined. Consider setting an expiration period to manage the growth of VPC Flow Logs. For more information,

see: http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/SettingLogRetention.html

References:

1. CCE-79202-8
2. CIS CSC v6.0 #6.5, #12.9

4.4 Ensure the default security group restricts all traffic (Scored)

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.

Rationale:
Configuring the default security group to restrict all traffic will encourage least privilege security group development and mindful placement of AWS resource into security groups which will in-turn reduce the exposure of those resources.

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:
    1. Select the `default` security group
    2. Click the `Inbound Rules` tab
    3. Ensure no rule exist
    4. Click the `Outbound Rules` tab
    5. Ensure no rules exist

Remediation:
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

References:

1. CCE-79201-0
2. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html
3. CIS CSC v6.0 #9.2

Control		Set Correctly	
		Yes	No
1	**Identity and Access Management**		
1.1	Avoid the use of the "root" account (Not Scored)	☐	☐
1.2	Ensure multi-factor authentication (MFA) is enabled for all IAM users that have a password (Scored)	☐	☐
1.3	Ensure credentials unused for 90 days or greater are disabled (Scored)	☐	☐
1.4	Ensure access keys are rotated every 90 days or less (Scored)	☐	☐
1.5	Ensure IAM password policy requires at least one uppercase letter (Scored)	☐	☐
1.6	Ensure IAM password policy require at least one lowercase letter (Scored)	☐	☐
1.7	Ensure IAM password policy require at least one symbol (Scored)	☐	☐
1.8	Ensure IAM password policy require at least one number (Scored)	☐	☐
1.9	Ensure IAM password policy requires minimum length of 14 or greater (Scored)	☐	☐
1.10	Ensure IAM password policy prevents password reuse (Scored)	☐	☐
1.11	Ensure IAM password policy expires passwords within 90 days or less (Scored)	☐	☐
1.12	Ensure no root account access key exists (Scored)	☐	☐
1.13	Ensure hardware MFA is enabled for the "root" account (Scored)	☐	☐
1.14	Ensure security questions are registered in the AWS account (Not Scored)	☐	☐
1.15	Ensure IAM policies are attached only to groups or roles (Scored)	☐	☐
2	**Logging**		
2.1	Ensure CloudTrail is enabled in all regions (Scored)	☐	☐
2.2	Ensure CloudTrail log file validation is enabled (Scored)	☐	☐
2.3	Ensure the S3 bucket CloudTrail logs to is not publicly accessible (Scored)	☐	☐
2.4	Ensure CloudTrail trails are integrated with CloudWatch Logs (Scored)	☐	☐
2.5	Ensure AWS Config is enabled in all regions (Scored)	☐	☐
2.6	Ensure S3 Bucket Access Logging is enabled for all buckets (Scored)	☐	☐
2.7	Ensure CloudTrail logs are encrypted at rest using KMS CMKs (Scored)	☐	☐
2.8	Ensure rotation for customer created CMKs is enabled (Scored)	☐	☐
3	**Monitoring**		
3.1	Ensure a log metric filter and alarm exist for unauthorized API calls (Scored)	☐	☐
3.2	Ensure a log metric filter and alarm exist for Management Console sign-in without MFA (Scored)	☐	☐

3.3	Ensure a log metric filter and alarm exist for usage of "root" account (Scored)	☐	☐
3.4	Ensure a log metric filter and alarm exist for IAM policy changes (Scored)	☐	☐
3.5	Ensure a log metric filter and alarm exist for CloudTrail configuration changes (Scored)	☐	☐
3.6	Ensure a log metric filter and alarm exist for AWS Management Console authorization failures (Scored)	☐	☐
3.7	Ensure a log metric filter and alarm exist for disabling or scheduled deletion of customer created CMKs (Scored)	☐	☐
3.8	Ensure a log metric filter and alarm exist for S3 bucket policy changes (Scored)	☐	☐
3.9	Ensure a log metric filter and alarm exist for AWS Config configuration changes (Scored)	☐	☐
3.10	Ensure a log metric filter and alarm exist for security group changes (Scored)	☐	☐
3.11	Ensure a log metric filter and alarm exist for changes to Network Access Control Lists (NACL) (Scored)	☐	☐
3.12	Ensure a log metric filter and alarm exist for changes to network gateways (Scored)	☐	☐
3.13	Ensure a log metric filter and alarm exist for route table changes (Scored)	☐	☐
3.14	Ensure a log metric filter and alarm exist for VPC changes (Scored)	☐	☐
3.15	Ensure security contact information is registered (Scored)	☐	☐
3.16	Ensure appropriate subscribers to each SNS topic (Not Scored)	☐	☐
4	**Networking**		
4.1	Ensure no security groups allow ingress from 0.0.0.0/0 to port 22 (Scored)	☐	☐
4.2	Ensure no security groups allow ingress from 0.0.0.0/0 to port 3389 (Scored)	☐	☐
4.3	Ensure VPC Flow Logging is Enabled in all Applicable Regions (Scored)	☐	☐
4.4	Ensure the default security group restricts all traffic (Scored)	☐	☐

Appendix: Change History

Date	Version	Changes for this version
02-29-2016	1.0.0	Initial Release