CIS Amazon Web Services Foundations

v1.1.0 - 11-09-2016

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:

Contributor
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
Adam Montville

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 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 commands will provide ad-hoc means for determining the last time the root account was used:

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

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

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

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

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

Remediation:

Perform the following to enable MFA:

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.

   IAM generates and displays configuration information for the virtual MFA device, including a QR code graphic. The graphic is a representation of the 'secret configuration key' that is available for manual entry on devices that do not support QR codes.

6. Open your virtual MFA application. (For a list of apps that you can use for hosting virtual MFA devices, see Virtual MFA Applications.) If the virtual MFA application supports multiple accounts (multiple virtual MFA devices), choose the option to create a new account (a new virtual MFA device).
7. Determine whether the MFA app supports QR codes, and then do one of the following:
   o 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.
   o 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.

Forced IAM User Self-Service Remediation

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

[How to Delegate Management of Multi-Factor Authentication to AWS IAM Users](#)

References:

1. http://tools.ietf.org/html/rfc6238
2. http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa.html
3. CCE-78901-6
4. 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 CLI

1. Run the following commands:

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

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

Via CLI

```
aws iam update-account-password-policy --require-uppercase-characters
```

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

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 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:

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

Via CLI

```
aws iam update-account-password-policy --require-lowercase-characters
```

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

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 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"

Via CLI

```
aws iam update-account-password-policy --require-symbols
```

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

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 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:

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 number"
5. Click "Apply password policy"

Via CLI

```
aws iam update-account-password-policy --require-numbers
```

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

References:

1. CCE-78906-5

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

Via CLI

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

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

References:

1. CCE-78907-3
2. 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 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`

Via CLI

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

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

References:

1. CCE-78908-1

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 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):" to 90 or less

Via CLI

```
aws iam update-account-password-policy --max-password-age 90
```

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

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 CLI

1. Run the following commands:

```
aws iam generate-credential-report
aws iam get-credential-report --query 'Content' --output text | base64 -d | cut
-d, -f1,9,14 | grep -B1 '<root_account>'
```

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 *<Root_Account_Name>* at the top right 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 MFA is enabled for the "root" account (Scored)

Profile Applicability:

- Level 1

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.

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

Rationale:

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

Audit:

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

1. Run the following command:

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

2. Ensure the AccountMFAEnabled property is set to 1

Remediation:

Perform the following to establish MFA for the root account:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

Note: to manage MFA devices for the root AWS account, you must use your root account credentials to sign in to AWS. You cannot manage MFA devices for the root account using other credentials.

2. Choose `Dashboard`, and under `Security Status`, expand `Activate MFA` on your root account.
3. Choose `Activate MFA`
4. In the wizard, choose `A virtual MFA` device and then choose `Next Step`.
5. IAM generates and displays configuration information for the virtual MFA device, including a QR code graphic. The graphic is a representation of the 'secret configuration key' that is available for manual entry on devices that do not support QR codes.
6. Open your virtual MFA application. (For a list of apps that you can use for hosting virtual MFA devices, see <u>Virtual MFA Applications</u>.) 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:
   o 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.
   o 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.

1. In the Manage MFA Device wizard, in the Authentication Code 1 box, type the one-time password that currently appears in the virtual MFA device. Wait up to 30 seconds for the device to generate a new one-time password. Then type the second one-time password into the Authentication Code 2 box. Choose Active Virtual MFA.

References:

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

1.14 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. For Level 2, 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.

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

Link to order AWS compatible hardware MFA device: http://onlinenoram.gemalto.com/

Audit:

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

1. Run the following command to list all virtual MFA devices:

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

2. If the output contains one MFA with the following Serial Number, it means the MFA is virtual, not hardware and the account is not compliant with this recommendation:

 "SerialNumber": "arn:aws:iam::$<aws\_account\_number>$:mfa/root-account-mfa-device"

Remediation:

Perform the following to establish a hardware MFA for the root 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
3. Order Hardware MFA: http://onlinenoram.gemalto.com/
4. http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_enable_virtual.html
5. http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_enable_physical.html#enable-hw-mfa-for-root

1.15 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 *<Root_Account_Name>*
3. Click on the *<Root_Account_Name>*
4. From the drop-down menu Click `My Account`
5. In the `Configure Security Challenge Questions` section on the `Personal Information` page, configure three security challenge questions.
6. Click `Save questions`.

Remediation:

Perform the following in the AWS Management Console:

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

- o Click on the `Answer` section
- o 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

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

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

1.17 Enable detailed billing (Scored)

Profile Applicability:

- Level 1

Description:

Enable Detailed Billing to cause the generation of a log record for every event or hourly ongoing activity which incurs cost in an AWS account. These records are aggregated into CSV files of hourly records, and written to an S3 bucket. A CSV (Comma Separated Values) file of billing records is written at least every 24 hours; writing of files is often more frequent.

Rationale:

Detailed Billing records can be used as an overview of AWS activity across the whole of an account, in addition to per-Region CloudTrail, Config and other service-specific JSON-based logs. Billing records can be graphed over time using the Cost Explorer tool, and budgeting alerts can be configured on billing records and pushed to SNS in the event of spend over time, or predicted spend at current rate, going above a customer-set threshold - this can be used as a simple means of detecting anomalous utilisation of AWS resources and thereby triggering investigation activities. Billing records can also be broken out by tag, which can serve as a starting point in identifying which part of the environment, or organisation, the anomalous activity is occurring in.

Audit:

There is currently no AWS CLI support for this operation, so it is necessary to use the Management Console.

As a user with IAM permission to read billing information (aws-portal:ViewBilling):

1. Sign in to the AWS Management Console and open the Billing and Cost Management console at https://console.aws.amazon.com/billing/home#/.
2. On the navigation pane, choose Preferences.
3. Verify whether the "Receive Billing Reports" check box is ticked. If it is not, billing reports are not being generated.

Remediation:

There is currently no AWS CLI support for this operation, so it is necessary to use the Management Console.

As a user with IAM permission to read and write billing information (aws-portal:*Billing):

- Sign in to the AWS Management Console and open the Billing and Cost Management console at https://console.aws.amazon.com/billing/home#/.
- On the navigation pane, choose Preferences.
- Select the Receive Billing Reports check box.
- Designate the Amazon S3 bucket *<S3_billing_bucket>* where you want AWS to publish your detailed billing reports.
- Ensure that policy allows read access only to appropriate groups of users (finance, auditors, etc). For appropriate groups in IAM who you want to have read access, include the following policy element:

```
"Statement":[

  {

    "Effect":"Allow",

    "Action":[

      "s3:GetObject",

      "s3:GetObjectVersion",

   "s3:GetBucketLocation"

    ],

    "Resource":"arn:aws:s3:::<S3_billing_bucket>/*"

  }

]
```

- After your S3 bucket has been verified, under Report, select the check boxes for the reports that you want to receive.
- Choose Save preferences
- Detailed billing reports can take up to 24 hours to start being generated. Wait >24 hours, and examine your designated S3 bucket to verify that files with names of the form (eg) <AWS account number>-<aws-billing-detailed-line-items-with-resources-and-tags-yyyy-mm>.csv.zip are being generated.

References:

1. https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/billing-what-is.html
2. https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/billing-permissions-ref.html#billing-example-policies

1.18 Ensure IAM Master and IAM Manager roles are active (Scored)

Profile Applicability:

- Level 1

Description:

Ensure IAM Master and IAM Manager roles are in place for IAM administration and assignment of administrative permissions for other services to other roles.

An IAM role is conceptually "a container of permissions resembling a user account which cannot be directly logged into, but which must instead be assumed from an existing user account which has appropriate permissions to do so", in the manner of roles in Unix Role-Based Access Control (RBAC). In AWS, roles can also be assigned to EC2 instances and Lambda functions.

Control over IAM, which is also defined and mediated by a number of fine-grained permissions, should be divided between a number of roles, such that no individual user in a production account has full control over IAM.

Rationale:

IAM is the principal point of control for service configuration access, and "control over IAM" means "control over the configuration of all other assets in the AWS account". Therefore it is recommended that control of this degree of security criticality should be divided among multiple individuals within an organisation, in a manner such that no individual retains enough control over IAM to "rewrite themselves to root".

Roles are recommended for security-sensitive capabilities, as the act of assuming a role generates a set of ephemeral credentials using the Security Token Service (STS) and these credentials - being a token, an AWS Access Key and an AWS Secret Access Key - are needed to make API calls in the context of the role. STS credentials expire after a configurable period (default 12 hours, minimum 15 minutes, maximum 36 hours), and this reduces the risk of engineers hard-wiring these keys into code, and therefore further reduces the risk of the keys being mishandled.

The current recommendation is to divide account and permission configuration permissions between 2 roles, which are:

IAM Master: creates users, groups and roles; assigns permissions to roles
IAM Manager: assigns users and roles to groups

In this model, IAM Master and IAM Manager must work together in a 2-person rule manner, in order for a user to gain access to a permission.

Audit:

Using the Amazon unified CLI, from a user or role which has the iam:ListRoles and iam:GetRolePolicy permissions:

List the configured roles:

```
aws iam list-roles --query "Roles[*].{RoleName:RoleName, Arn:Arn}"
```
The output should contain entries with "RoleName": "*<iam_manager_role_name>*" and "Rolename": "*<iam_master_role_name>*"

Examine the permissions associated with each of these roles:

```
aws iam get-role-policy —role-name <iam_manager_role_name>
aws iam get-role-policy —role-name <iam_master_role_name>
```

The *<iam_master_role_name>* role should include the following Actions with an Allow effect:

iam: AttachRolePolicy
iam:CreateGroup
iam:CreatePolicy
iam:CreatePolicyVersion
iam:CreateRole
iam:CreateUser
iam:DeleteGroup
iam:DeletePolicy
iam:DeletePolicyVersion
iam:DeleteRole
iam:DeleteRolePolicy
iam:DeleteUser
iam:PutRolePolicy
iam:GetPolicy
iam:GetPolicyVersion
iam:GetRole
iam:GetRolePolicy

iam:GetUser
iam:GetUserPolicy
iam:ListEntitiesForPolicy
iam:ListGroupPolicies
iam:ListGroups
iam:ListGroupsForUser
iam:ListPolicies
iam:ListPoliciesGrantingServiceAccess
iam:ListPolicyVersions
iam:ListRolePolicies
iam:ListAttachedGroupPolicies
iam:ListAttachedRolePolicies
iam:ListAttachedUserPolicies
iam:ListRoles
iam:ListUsers

...and the following Actions with a Deny effect:

iam:AddUserToGroup
iam:AttachGroupPolicy
iam:DeleteGroupPolicy
iam:DeleteUserPolicy
iam:DetachGroupPolicy
iam:DetachRolePolicy
iam:DetachUserPolicy
iam:PutGroupPolicy
iam:PutUserPolicy
iam:RemoveUserFromGroup
iam:UpdateGroup
iam:UpdateAssumeRolePolicy
iam:UpdateUser

The *<iam_manager_role_name>* role should include the following Actions with an Allow
effect:

iam:AddUserToGroup
iam:AttachGroupPolicy
iam:DeleteGroupPolicy
iam:DeleteUserPolicy
iam:DetachGroupPolicy

iam:DetachRolePolicy
iam:DetachUserPolicy
iam:PutGroupPolicy
iam:PutUserPolicy
iam:RemoveUserFromGroup
iam:UpdateGroup
iam:UpdateAssumeRolePolicy
iam:UpdateUser
iam:GetPolicy
iam:GetPolicyVersion
iam:GetRole
iam:GetRolePolicy
iam:GetUser
iam:GetUserPolicy
iam:ListEntitiesForPolicy
iam:ListGroupPolicies
iam:ListGroups
iam:ListGroupsForUser
iam:ListPolicies
iam:ListPoliciesGrantingServiceAccess
iam:ListPolicyVersions
iam:ListRolePolicies
iam:ListAttachedGroupPolicies
iam:ListAttachedRolePolicies
iam:ListAttachedUserPolicies
iam:ListRoles
iam:ListUsers

...and the following Actions with a Deny effect:

iam: AttachRolePolicy
iam:CreateGroup
iam:CreatePolicy
iam:CreatePolicyVersion
iam:CreateRole
iam:CreateUser
iam:DeleteGroup
iam:DeletePolicy
iam:DeletePolicyVersion
iam:DeleteRole

iam:DeleteRolePolicy
iam:DeleteUser
iam:PutRolePolicy

Other iam:* Actions may be included in these policies as needed.

Both policies should also be limited by a Condition that MFA authentication is in effect, by containing:

"Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}

in the Allow effect section (provided IAM Federation has not been configured).

Each role needs to be assumable by at least one user or group:

```
aws iam get-role —role-name <iam_manager_role_name>
aws iam get-role —role-name <iam_master_role_name>
```

should display the AssumeRolePolicyDocument indicating which users and groups are able to assume the roles. No user or group should be able to assume both roles.

Remediation:

Using the Amazon unified CLI, from a user or role which has the iam:CreateRole, iam:CreatePolicy and iam:PutRolePolicy permissions:

```
aws iam create-role --role-name <iam_manager_role_name>
aws iam create-role --role-name <iam_master_role_name>
```

```
aws iam put-role-policy --role-name <iam_manager_role_name> --policy-name
<iam_manager_permissions_policy> --policy-document file://IAM-Manager-policy.json
aws iam put-role-policy --role-name <iam_master_role_name> --policy-name
<iam_master_permissions_policy> --policy-document file://IAM-Master-policy.json
```

where IAM-Master-policy.json contains:

```
{

    "Version": "2012-10-17",

    "Statement": [{
```

```
    "Action": [

        "iam:CreateGroup",

    "iam:CreatePolicy",

    "iam:CreatePolicyVersion",

    "iam:CreateRole",

    "iam:CreateUser",

    "iam:DeleteGroup",

    "iam:DeletePolicy",

    "iam:DeletePolicyVersion",

    "iam:DeleteRole",

    "iam:DeleteRolePolicy",

    "iam:DeleteUser",

    "iam:PutRolePolicy",

    "iam:GetPolicy",

    "iam:GetPolicyVersion",

    "iam:GetRole",

    "iam:GetRolePolicy",

    "iam:GetUser",

    "iam:GetUserPolicy",

    "iam:ListEntitiesForPolicy",

    "iam:ListGroupPolicies",

    "iam:ListGroups",

    "iam:ListGroupsForUser",
```

"iam:ListPolicies",

"iam:ListPoliciesGrantingServiceAccess",

"iam:ListPolicyVersions",

"iam:ListRolePolicies",

"iam:ListAttachedGroupPolicies",

"iam:ListAttachedRolePolicies",

"iam:ListAttachedUserPolicies",

"iam:ListRoles",

"iam:ListUsers"

    ],

    "Effect": "Allow",

    "Resource": "*",

    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}

  }],

    "Action": [

"iam:AddUserToGroup",

"iam:AttachGroupPolicy",

"iam:DeleteGroupPolicy",

"iam:DeleteUserPolicy",

"iam:DetachGroupPolicy",

"iam:DetachRolePolicy",

"iam:DetachUserPolicy",

"iam:PutGroupPolicy",

```
"iam:PutUserPolicy",

"iam:RemoveUserFromGroup",

"iam:UpdateGroup",

"iam:UpdateAssumeRolePolicy",

"iam:UpdateUser"

     ],

     "Effect": "Deny",

     "Resource": "*"

   }]

}
```

and where IAM-Manager-policy.json contains:

```
{

   "Version": "2012-10-17",

   "Statement": [{

     "Action": [

"iam:AddUserToGroup",

"iam:AttachGroupPolicy",

"iam:DeleteGroupPolicy",

"iam:DeleteUserPolicy",

"iam:DetachGroupPolicy",

"iam:DetachRolePolicy",
```

"iam:DetachUserPolicy",

"iam:PutGroupPolicy",

"iam:PutUserPolicy",

"iam:RemoveUserFromGroup",

"iam:UpdateGroup",

"iam:UpdateAssumeRolePolicy",

"iam:UpdateUser",

"iam:GetPolicy",

"iam:GetPolicyVersion",

"iam:GetRole",

"iam:GetRolePolicy",

"iam:GetUser",

"iam:GetUserPolicy",

"iam:ListEntitiesForPolicy",

"iam:ListGroupPolicies",

"iam:ListGroups",

"iam:ListGroupsForUser",

"iam:ListPolicies",

"iam:ListPoliciesGrantingServiceAccess",

"iam:ListPolicyVersions",

"iam:ListRolePolicies",

"iam:ListAttachedGroupPolicies",

"iam:ListAttachedRolePolicies",

```
"iam:ListAttachedUserPolicies",

"iam:ListRoles",

"iam:ListUsers"

    ],

    "Effect": "Allow",

    "Resource": "*",

    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}

  }],

    "Action": [

      "iam:CreateGroup",

"iam:CreatePolicy",

"iam:CreatePolicyVersion",

"iam:CreateRole",

"iam:CreateUser",

"iam:DeleteGroup",

"iam:DeletePolicy",

"iam:DeletePolicyVersion",

"iam:DeleteRole",

"iam:DeleteRolePolicy",

"iam:DeleteUser",

"iam:PutRolePolicy"

    ],

    "Effect": "Deny",
```

```
    "Resource": "*"

  }]

}
```

Note that each of IAM-Manager-policy.json and IAM-Master-policy.json can contain other iam:* permissions in either Allow or Deny Action lists, depending on what other requirements are in place in the account.

Each of these roles needs to be assumable by a different user or group.

For appropriate users or groups (groups are recommended):

```
aws iam put-user-policy --user-name <iam_user> --policy-name
<assume_iam_master_role_policy> --policy-document file://Assume-IAM-Master.json
aws iam put-user-policy --user-name <iam_user> --policy-name
<assume_iam_manager_role_policy> --policy-document file://Assume-IAM-Manager.json
```

or

```
aws iam put-group-policy --group-name <iam_group>  --policy-name
<assume_iam_master_role_policy> --policy-document file://Assume-IAM-Master.json
aws iam put-group-policy --group-name <iam_group>  --policy-name
<assume_iam_manager_role_policy> --policy-document file://Assume-IAM-Manager.json
```

where Assume-IAM-Master.json is:

```
{

  "Version": "2012-10-17",

  "Statement": {

    "Effect": "Allow",

    "Action": "sts:AssumeRole",
```

```
    "Resource": "arn:aws:iam::<aws_account_number>:role/<iam_master_role_name>"

 }

}
```

and Assume-IAM-Manager.json is:

```
{

  "Version": "2012-10-17",

  "Statement": {

    "Effect": "Allow",

    "Action": "sts:AssumeRole",

    "Resource": "arn:aws:iam::<aws_account_number>:role/<iam_manager_role_name>"

 }

}
```

References:

1. https://docs.aws.amazon.com/IAM/latest/UserGuide/list_iam.html
2. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_configure
   -api-require.html

1.19 Maintain current contact details (Scored)

Profile Applicability:

- Level 1

Description:

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

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

Rationale:

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

Audit:

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

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

Remediation:

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

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

References:

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

1.20 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

1.21 Ensure IAM instance roles are used for AWS resource access from instances (Not Scored)

Profile Applicability:

- Level 2

Description:

AWS access from within AWS instances can be done by either encoding AWS keys into AWS API calls or by assigning the instance to a role which has an appropriate permissions policy for the required access. "AWS Access" means accessing the APIs of AWS in order to access AWS resources or manage AWS account resources.

Rationale:

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

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

Audit:

Whether an Instance Is Associated With a Role

For instances that are known to perform AWS actions, ensure that they belong to an instance role that has the necessary permissions:

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

Whether an Instance Contains Embedded Credentials

On the instance that is known to perform AWS actions, audit all scripts and environment variables to ensure that none of them contain AWS credentials.

Whether an Instance Application Contains Embedded Credentials

Applications that run on an instance may also have credentials embedded. This is a bad practice, but even worse if the source code is stored in a public code repository such as github. When an application contains credentials can be determined by eliminating all other sources of credentials and if the application can still access AWS resources - it likely contains embedded credentials. Another method is to examine all source code and configuration files of the application.

Remediation:

IAM roles can only be associated at the launch of an instance. To remediate an instance to add it to a role you must create a new instance.

If the instance has no external dependencies on it's current private ip or public addresses are elastic IPs:

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

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

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

References:

1. http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html
2. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html
3. CIS CSC v6.0 #16.14 (someone please check the applicability of this for me)

1.22 Ensure a support role has been created to manage incidents with AWS Support (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

Using the Amazon unified command line interface:

- List IAM policies, filter for the 'AWSSupportAccess' managed policy, and note the "Arn" element value:
- `aws iam list-policies --query "Policies[?PolicyName == 'AWSSupportAccess']"`

- Check if the 'AWSSupportAccess' is attached to any IAM user, group or role:
- `aws iam list-entities-for-policy --policy-arn <iam_policy_arn>`

Remediation:

Using the Amazon unified command line interface:

- Create an IAM role for managing incidents with AWS:
  - Create a trust relationship policy document that allows *<iam_user>* to manage AWS incidents, and save it locally as /tmp/TrustPolicy.json:
  -
    ```
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "AWS": "<iam_user>"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
    ```

-
    - Create the IAM role using the above trust policy:
    - `aws iam create-role --role-name <aws_support_iam_role> --assume-role-policy-document file:///tmp/TrustPolicy.json`

-
    - Attach 'AWSSupportAccess' managed policy to the created IAM role:
    - `aws iam attach-role-policy --policy-arn <iam_policy_arn> --role-name <aws_support_iam_role>`

Impact:

All AWS Support plans include an unlimited number of account and billing support cases, with no long-term contracts.

Support billing calculations are performed on a per-account basis for all plans. Enterprise Support plan customers have the option to include multiple enabled accounts in an aggregated monthly billing calculation.

Monthly charges for the Business and Enterprise support plans are based on each month's AWS usage charges, subject to a monthly minimum, billed in advance.

References:

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

1.23 Do not setup access keys during initial user setup for all IAM users that have a console password (Not Scored)

Profile Applicability:

- Level 1

Description:

AWS console defaults the checkbox for creating access keys to enabled. This results in many access keys being generated unnecessarily. In addition to unnecessary credentials, it also generates unnecessary management work in auditing and rotating these keys.

Rationale:

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

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

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 onA User
5. Compare the user creation date to the key 1 creation date.
6. For any that match, the key was created during initial user setup.

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

Via the CLI

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

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

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

```
user,password_enabled,access_key_1_active,access_key_1_last_used_date,access_ke
y_2_active,access_key_2_last_used_date
elise,false,true,2015-04-16T15:14:00+00:00,false,N/A
brandon,true,true,N/A,false,N/A
rakesh,false,false,N/A,false,N/A
helene,false,true,2015-11-18T17:47:00+00:00,false,N/A
paras,true,true,2016-08-28T12:04:00+00:00,true,2016-03-04T10:11:00+00:00
anitha,true,true,2016-06-08T11:43:00+00:00,true,N/A
```

3. For any user having `access_key_last_used_date` set to `N/A`, ensure that access key is deleted .

Remediation:

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

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
   o Click on `Delete` for keys that were created at the same time as the user profile but have not been used.
7. As an IAM User
   o Click on `Delete` for keys that were created at the same time as the user profile but have not been used.

Via CLI

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

1.24 Ensure IAM policies that allow full "*:*" administrative privileges are not created (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

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

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

Audit:

Perform the following to determine what policies are created:

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

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

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

   ```
   aws iam get-policy-version --policy-arn <policy_arn> --version <policy_version>
   --query "PolicyVersion.Document.Statement[?Effect == 'Allow' &&
   contains(Resource, '*') && contains (Action, '*')]"
   ```

3. If the output of the command returns any policies, it's not compliant.

Remediation:

Using the GUI, perform the following to detach the policy that has full administrative privileges:

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

Using the CLI, perform the following to detach the policy that has full administrative privileges as found in the audit step:

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

```
1. aws iam list-entities-for-policy --policy-arn <policy_arn>
```

2. Detach the policy from all IAM Users:

```
1. aws iam detach-user-policy --user-name <iam_user> --policy-arn <policy_arn>
```

3. Detach the policy from all IAM Groups:

```
1. aws iam detach-group-policy --group-name <iam_group> --policy-arn <policy_arn>
```

4. Detach the policy from all IAM Roles:

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

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
4. http://docs.aws.amazon.com/cli/latest/reference/iam/index.html#cli-aws-iam

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 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 CLI

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

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 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 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 <trail_arn> --start-time <start_time> --end-
time <end_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 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 <s3_bucket_for_cloudtrail> --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 <s3_bucket_for_cloudtrail> --query
'Grants[?Grantee.URI==`http://acs.amazonaws.com/groups/global/Authenticated
Users`]'
```

4. Get the S3 Bucket Policy

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

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

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 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 <trail_name>
```

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

Remediation:

Perform the following to establish the prescribed state:

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 CLI

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

Impact:

Note: By default, CloudWatch Logs will store Logs indefinitely unless a specific retention period is defined for the log group. When choosing the number of days to retain, keep in mind the average days it takes an organization to realize they have been breached is 210 days (at the time of this writing). Since additional time is required to research a breach, a

minimum 365 day retention policy allows time for detection and research. You may also wish to archive the logs to a cheaper storage service rather than simply deleting them. See the following AWS resource to manage CloudWatch Logs retention periods:

1. 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 CLI

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

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:`*<s3_bucket_for_cloudtrail>* 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 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 CLI

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

Impact:

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

References:

1. CIS CSC v6.0 #13.1: Perform an assessment of data to identify sensitive information.
2. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/encrypting-cloudtrail-log-files-with-aws-kms.html
3. http://docs.aws.amazon.com/kms/latest/developerguide/create-keys.html
4. CIS CSC v6.0 #6: Maintenance, Monitoring, and Analysis of Audit Logs
5. CCE-78919-8

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 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 <kms_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 CLI

1. Run the following command to enable key rotation:

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

References:

1. https://aws.amazon.com/kms/pricing/
2. 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 *<cloudtrail_log_group_name>* value associated with
`CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<aws_account_number>:log-group:<cloudtrail_log_group_name>:*"
```
3. Get a list of all associated metric filters for this *<cloudtrail_log_group_name>*:

```
aws logs describe-metric-filters --log-group-name "<cloudtrail_log_group_name>"
```
4. Ensure the output from the above command contains the following:

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

5. Note the *<unauthorized_api_calls_metric>* value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the *<unauthorized_api_calls_metric>* captured in step 5.

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

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 <sns_topic_arn>
```

Remediation:

Perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for unauthorized API calls and the *<cloudtrail_log_group_name>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name
<unauthorized_api_calls_metric> --metric-transformations
metricName=<unauthorized_api_calls_metric>,metricNamespace='CISBenchmark',metricVal
ue=1 --filter-pattern '{ ($.errorCode = "*UnauthorizedOperation") || ($.errorCode =
"AccessDenied*") }'
```

Note: You can choose your own `metricName` and `metricNamespace` strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.

2. Create an SNS topic that the alarm will notify

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

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

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

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

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

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

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

Note: set the `period` and `threshold` to values that fit your organization.

Impact:

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

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

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

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 *<cloudtrail_log_group_name>* value associated with

`CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<aws_account_number>:log-group:<cloudtrail_log_group_name>:*"
```

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

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

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

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

5. Note the `<no_mfa_console_signin_metric>` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `<no_mfa_console_signin_metric>` captured in step 5.

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

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 <sns_topic_arn>
```

Remediation:

Perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for AWS Management Console sign-in without MFA and the *<cloudtrail_log_group_name>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name
<no_mfa_console_signin_metric> --metric-transformations
metricName=<no_mfa_console_signin_metric>,metricNamespace='CISBenchmark',metricValu
e=1 --filter-pattern '{ ($.eventName = "ConsoleLogin") &&
($.additionalEventData.MFAUsed != "Yes") }'
```

Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.

2. Create an SNS topic that the alarm will notify

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

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

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

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

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

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

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

References:

1. [http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/viewing _metrics_with_cloudwatch.html](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 *<cloudtrail_log_group_name>* value associated with

`CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<aws_account_number>:log-group:<cloudtrail_log_group_name>:*"
```

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

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

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

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

5. Note the `<root_usage_metric>` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `<root_usage_metric>` captured in step 5.

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

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 <sns_topic_arn>
```

Remediation:

Perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for "Root" account usage and the *<cloudtrail_log_group_name>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name
<root_usage_metric> --metric-transformations
metricName=<root_usage_metric>,metricNamespace='CISBenchmark',metricValue=1 --
filter-pattern '{ $.userIdentity.type = "Root" && $.userIdentity.invokedBy NOT EXISTS
&& $.eventType != "AwsServiceEvent" }'
```

Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.

2. Create an SNS topic that the alarm will notify

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

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

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

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

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

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

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

References:

1. CCE-79188-9
2. 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 *<cloudtrail_log_group_name>* value associated with

`CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<aws_account_number>:log-group:<cloudtrail_log_group_name>:*"
```

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

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

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 `<iam_changes_metric>` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `<iam_changes_metric>` captured in step 5.

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

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 <sns_topic_arn>
```

Remediation:

Perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for IAM Policy changes and the *<cloudtrail_log_group_name>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name
<iam_changes_metric> --metric-transformations
metricName=<iam_changes_metric>,metricNamespace='CISBenchmark',metricValue=1 --
filter-pattern
'{($.eventName=DeleteGroupPolicy)||($.eventName=DeleteRolePolicy)||($.eventName=Delete
UserPolicy)||($.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)}'
```

Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.

2. Create an SNS topic that the alarm will notify

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

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

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

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

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

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

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

References:

1. CCE-79189-7

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 *<cloudtrail_log_group_name>* value associated with

`CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<aws_account_number>:log-group:<cloudtrail_log_group_name>:*"
```

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

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

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 `<cloudtrail_cfg_changes_metric>` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `<cloudtrail_cfg_changes_metric>` captured in step 5.

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

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 <sns_topic_arn>
```

Remediation:

Perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for Cloudtrail configuration changes and the *<cloudtrail_log_group_name>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name
<cloudtrail_cfg_changes_metric> --metric-transformations
metricName=<cloudtrail_cfg_changes_metric>,metricNamespace='CISBenchmark',metricVal
ue=1 --filter-pattern '{ ($.eventName = CreateTrail) || ($.eventName = UpdateTrail) ||
($.eventName = DeleteTrail) || ($.eventName = StartLogging) || ($.eventName =
StopLogging) }'
```

Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.

2. Create an SNS topic that the alarm will notify

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

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

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

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

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

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

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

References:

1. CCE-79190-5

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 *<cloudtrail_log_group_name>* value associated with

`CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<aws_account_number>:log-group:<cloudtrail_log_group_name>:*"
```
3. Get a list of all associated metric filters for this *<cloudtrail_log_group_name>*:

```
aws logs describe-metric-filters --log-group-name "<cloudtrail_log_group_name>"
```
4. Ensure the output from the above command contains the following:

```
"filterPattern": "{ ($.eventName = ConsoleLogin) && ($.errorMessage = \"Failed
authentication\") }"
```
5. Note the `<console_signin_failure_metric>` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `<console_signin_failure_metric>` captured in step 5.

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

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 <sns_topic_arn>
```


Remediation:

Perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for AWS Management Console authentication failures and the *<cloudtrail_log_group_name>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name
<console signin failure metric> --metric-transformations
metricName=<console_signin_failure_metric>,metricNamespace='CISBenchmark',metricVal
ue=1 --filter-pattern '{ ($.eventName = ConsoleLogin) && ($.errorMessage = ""Failed
authentication"") }'
```
Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.

2. Create an SNS topic that the alarm will notify

```
aws sns create-topic --name <sns_topic_name>
```
Note: you can execute this command once and then re-use the same topic for all monitoring alarms.

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

```
aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --
notification-endpoint <sns_subscription_endpoints>
```
Note: you can execute this command once and then re-use the SNS subscription for all monitoring alarms.

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

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

References:

1. CCE-79191-3

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 *<cloudtrail_log_group_name>* value associated with

`CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<aws_account_number>:log-group:<cloudtrail_log_group_name>:*"
```

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

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

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

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

5. Note the `<disable_or_delete_cmk_metric>` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `<disable_or_delete_cmk_metric>` captured in step 5.

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

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 <sns_topic_arn>
```

Remediation:

Perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for disabled or scheduled for deletion CMK's and the *<cloudtrail_log_group_name>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name
<disable_or_delete_cmk_metric> --metric-transformations
metricName=<disable_or_delete_cmk_metric>,metricNamespace='CISBenchmark',metricValu
e=1 --filter-pattern '{($.eventSource = kms.amazonaws.com) &&
(($.eventName=DisableKey)||($.eventName=ScheduleKeyDeletion))}'
```
Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.

2. Create an SNS topic that the alarm will notify

```
aws sns create-topic --name <sns_topic_name>
```
Note: you can execute this command once and then re-use the same topic for all monitoring alarms.

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

```
aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --
notification-endpoint <sns_subscription_endpoints>
```
Note: you can execute this command once and then re-use the SNS subscription for all monitoring alarms.

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

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

References:

1. CCE-79192-1

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 *<cloudtrail_log_group_name>* value associated with

`CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<aws_account_number>:log-group:<cloudtrail_log_group_name>:*"
```

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

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

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 `<s3_bucket_policy_changes_metric>` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `<s3_bucket_policy_changes_metric>` captured in step 5.

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

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 <sns_topic_arn>
```

Remediation:

Perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for S3 Bucket Policy changes and the *<cloudtrail_log_group_name>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name
<s3_bucket_policy_changes_metric> --metric-transformations
metricName=<s3_bucket_policy_changes_metric>,metricNamespace='CISBenchmark',metric
Value=1 --filter-pattern '{ ($.eventSource = s3.amazonaws.com) && (($.eventName =
PutBucketAcl) || ($.eventName = PutBucketPolicy) || ($.eventName = PutBucketCors) ||
($.eventName = PutBucketLifecycle) || ($.eventName = PutBucketReplication) ||
($.eventName = DeleteBucketPolicy) || ($.eventName = DeleteBucketCors) || ($.eventName
= DeleteBucketLifecycle) || ($.eventName = DeleteBucketReplication)) }'
```
Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.

2. Create an SNS topic that the alarm will notify

```
aws sns create-topic --name <sns_topic_name>
```
Note: you can execute this command once and then re-use the same topic for all monitoring alarms.

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

```
aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --
notification-endpoint <sns_subscription_endpoints>
```
Note: you can execute this command once and then re-use the SNS subscription for all monitoring alarms.

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

```
aws cloudwatch put-metric-alarm --alarm-name <s3_bucket_policy_changes_alarm> --
metric-name <s3_bucket_policy_changes_metric> --statistic Sum --period 300 --
threshold 1 --comparison-operator GreaterThanOrEqualToThreshold --evaluation-periods 1
--namespace 'CISBenchmark' --alarm-actions <sns_topic_arn>
```

References:

1. CCE-79193-9

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 *<cloudtrail_log_group_name>* value associated with

`CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<aws_account_number>:log-group:<cloudtrail_log_group_name>:*"
```

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

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

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 `<aws_config_changes_metric>` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `<aws_config_changes_metric>` captured in step 5.

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

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 <sns_topic_arn>
```

Remediation:

Perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for AWS Config changes and the *<cloudtrail_log_group_name>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name
<aws_config_changes_metric> --metric-transformations
metricName=<aws_config_changes_metric>,metricNamespace='CISBenchmark',metricValue=1
--filter-pattern '{($.eventSource = config.amazonaws.com) &&
(($.eventName=StopConfigurationRecorder)||($.eventName=DeleteDeliveryChannel)||($.even
tName=PutDeliveryChannel)||($.eventName=PutConfigurationRecorder))}'
```
Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.

2. Create an SNS topic that the alarm will notify

```
aws sns create-topic --name <sns_topic_name>
```
Note: you can execute this command once and then re-use the same topic for all monitoring alarms.

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

```
aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --
notification-endpoint <sns_subscription_endpoints>
```
Note: you can execute this command once and then re-use the SNS subscription for all monitoring alarms.

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

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

References:

1. CCE-79194-7

2. 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 *<cloudtrail_log_group_name>* value associated with

`CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<aws_account_number>:log-group:<cloudtrail_log_group_name>:*"
```

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

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

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 `<security_group_changes_metric>` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `<security_group_changes_metric>` captured in step 5.

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

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 <sns_topic_arn>
```

Remediation:

Perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for security groups changes and the *<cloudtrail_log_group_name>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name
<security_group_changes_metric> --metric-transformations
metricName=<security_group_changes_metric>,metricNamespace='CISBenchmark',metricVal
ue=1 --filter-pattern '{ ($.eventName = AuthorizeSecurityGroupIngress) || ($.eventName
= AuthorizeSecurityGroupEgress) || ($.eventName = RevokeSecurityGroupIngress) ||
($.eventName = RevokeSecurityGroupEgress) || ($.eventName = CreateSecurityGroup) ||
($.eventName = DeleteSecurityGroup)}'
```

Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.

2. Create an SNS topic that the alarm will notify

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

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

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

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

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

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

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

References:

1. CCE-79195-4

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 *<cloudtrail_log_group_name>* value associated with

`CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<aws_account_number>:log-group:<cloudtrail_log_group_name>:*"
```

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

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

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 `<nacl_changes_metric>` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `<nacl_changes_metric>` captured in step 5.

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

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 <sns_topic_arn>
```

Remediation:

Perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for NACL changes and the *<cloudtrail_log_group_name>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name
<nacl_changes_metric> --metric-transformations
metricName=<nacl_changes_metric>,metricNamespace='CISBenchmark',metricValue=1 --
filter-pattern '{ ($.eventName = CreateNetworkAcl) || ($.eventName =
CreateNetworkAclEntry) || ($.eventName = DeleteNetworkAcl) || ($.eventName =
DeleteNetworkAclEntry) || ($.eventName = ReplaceNetworkAclEntry) || ($.eventName =
ReplaceNetworkAclAssociation) }'
```
Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.

2. Create an SNS topic that the alarm will notify

```
aws sns create-topic --name <sns_topic_name>
```
Note: you can execute this command once and then re-use the same topic for all monitoring alarms.

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

```
aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --
notification-endpoint <sns_subscription_endpoints>
```
Note: you can execute this command once and then re-use the SNS subscription for all monitoring alarms.

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

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

References:

1. CCE-79196-2

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 *<cloudtrail_log_group_name>* value associated with

`CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<aws_account_number>:log-group:<cloudtrail_log_group_name>:*"
```

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

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

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 `<network_gw_changes_metric>` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `<network_gw_changes_metric>` captured in step 5.

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

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 <sns_topic_arn>
```


Remediation:

Perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for network gateways changes and the *<cloudtrail_log_group_name>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name
<network_gw_changes_metric> --metric-transformations
metricName=<network_gw_changes_metric>,metricNamespace='CISBenchmark',metricValue=1
--filter-pattern '{ ($.eventName = CreateCustomerGateway) || ($.eventName =
DeleteCustomerGateway) || ($.eventName = AttachInternetGateway) || ($.eventName =
CreateInternetGateway) || ($.eventName = DeleteInternetGateway) || ($.eventName =
DetachInternetGateway) }'
```

Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.

2. Create an SNS topic that the alarm will notify

```
aws sns create-topic --name <sns_topic_name>
```
Note: you can execute this command once and then re-use the same topic for all monitoring alarms.

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

```
aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --
notification-endpoint <sns_subscription_endpoints>
```
Note: you can execute this command once and then re-use the SNS subscription for all monitoring alarms.

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

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

References:

1. CCE-79197-0

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 *<cloudtrail_log_group_name>* value associated with

`CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<aws_account_number>:log-group:<cloudtrail_log_group_name>:*"
```

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

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

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 `<route_table_changes_metric>` value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the `<route_table_changes_metric>` captured in step 5.

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

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 <sns_topic_arn>
```

Remediation:

Perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for route table changes and the *<cloudtrail_log_group_name>* taken from audit step 2.

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

Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.

2. Create an SNS topic that the alarm will notify

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

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

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

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

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

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

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

References:

1. CCE-79198-8

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 *<cloudtrail_log_group_name>* value associated with

`CloudWatchLogsLogGroupArn`:

```
"arn:aws:logs:eu-west-1:<aws_account_number>:log-group:<cloudtrail_log_group_name>:*"
```
3. Get a list of all associated metric filters for this *<cloudtrail_log_group_name>*:

```
aws logs describe-metric-filters --log-group-name "<cloudtrail_log_group_name>"
```
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 *<vpc_changes_metric>* value associated with the `filterPattern` found in step 4.

6. Get a list of CloudWatch alarms and filter on the *<unauthorized_api_calls_metric>* captured in step 5.

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

Remediation:

Perform the following to setup the metric filter, alarm, SNS topic, and subscription:

1. Create a metric filter based on filter pattern provided which checks for VPC changes and the *<cloudtrail_log_group_name>* taken from audit step 2.

```
aws logs put-metric-filter --log-group-name <cloudtrail_log_group_name> --filter-name
<vpc_changes_metric> --metric-transformations
metricName=<vpc_changes_metric>,metricNamespace='CISBenchmark',metricValue=1 --
filter-pattern '{ ($.eventName = CreateVpc) || ($.eventName = DeleteVpc) ||
($.eventName = ModifyVpcAttribute) || ($.eventName = AcceptVpcPeeringConnection) ||
($.eventName = CreateVpcPeeringConnection) || ($.eventName =
DeleteVpcPeeringConnection) || ($.eventName = RejectVpcPeeringConnection) ||
($.eventName = AttachClassicLinkVpc) || ($.eventName = DetachClassicLinkVpc) ||
($.eventName = DisableVpcClassicLink) || ($.eventName = EnableVpcClassicLink) }'
```
Note: You can choose your own metricName and metricNamespace strings. Using the same metricNamespace for all Foundations Benchmark metrics will group them together.

2. Create an SNS topic that the alarm will notify

```
aws sns create-topic --name <sns_topic_name>
```
Note: you can execute this command once and then re-use the same topic for all monitoring alarms.

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

```
aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --
notification-endpoint <sns_subscription_endpoints>
```
Note: you can execute this command once and then re-use the SNS subscription for all monitoring alarms.

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

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

References:

1. CCE-79199-6

3.15 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 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`

Impact:

For updating an existing environment, care should be taken to ensure that administrators currently relying on an existing ingress from 0.0.0.0/0 have access to ports 22 and/or 3389 through another security group.

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`

Impact:

For updating an existing environment, care should be taken to ensure that administrators currently relying on an existing ingress from 0.0.0.0/0 have access to ports 22 and/or 3389 through another security group.

4.3 Ensure VPC flow logging is enabled in all VPCs (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 packet "Rejects" for VPCs.

Rationale:

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

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. `For Filter, select Reject`
8. Enter in a `Role` and `Destination Log Group`

9. Click `Create Log Flow`
10. Click on `CloudWatch Logs Group`

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

Impact:

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

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

References:

1. CCE-79202-8
2. CIS CSC v6.0 #6.5, #12.9
3. http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/flow-logs.html

4.4 Ensure the default security group of every VPC 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.

The default VPC in every region should have it's default security group updated to comply. Any newly created VPCs will automatically contain a default security group that will need remediation to comply with this recommendation.

NOTE: When implementing this recommendation, VPC flow logging is invaluable in determining the least privilege port access required by systems to work properly because it can log all packet acceptances and rejections occurring under the current security groups. This dramatically reduces the primary barrier to least privilege engineering - discovering the minimum ports required by systems in the environment. Even if the VPC flow logging recommendation in this benchmark is not adopted as a permanent security measure, it should be used during any period of discovery and engineering for least privileged security groups.

Rationale:

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

Audit:

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

Security Group State

1. Login to the AWS Management Console
   at https://console.aws.amazon.com/vpc/home
2. Repeat the next steps for all VPCs - including the default VPC in each AWS region:
3. In the left pane, click `Security Groups`
4. For each default security group, perform the following:
   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

Security Group Members

1. Login to the AWS Management Console
   at https://console.aws.amazon.com/vpc/home
2. Repeat the next steps for all default groups in all VPCs - including the default VPC in each AWS region:
3. In the left pane, click `Security Groups`
4. Copy the id of the default security group.
5. Change to the EC2 Management Console
   at https://console.aws.amazon.com/ec2/v2/home
6. In the filter column type 'Security Group ID : <security group id from #4>'

Remediation:

Security Group Members

Perform the following to implement the prescribed state:

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

Security Group State

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

5. Remove any inbound rules

Recommended:

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

Impact:

Implementing this recommendation in an existing VPC containing operating resources requires extremely careful migration planning as the default security groups are likely to be enabling many ports that are unknown. Enabling VPC flow logging (of accepts) in an existing environment that is know to be breach free will reveal the current pattern of ports being used for each instance to communicate successfully.

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

4.5 Ensure routing tables for VPC peering are "least access" (Not Scored)

Profile Applicability:

- Level 2

Description:

Once a VPC peering connection is estalished, routing tables must be updated to establish any connections between the peered VPCs. These routes can be as specific as desired - even peering a VPC to only a single host on the other side of the connection.

Rationale:

Being highly selective in peering routing tables is a very effective way of minimizing the impact of breach as resources outside of these routes are inaccessible to the peered VPC.

Audit:

Review routing tables of peered VPCs for whether they route all subnets of each VPC and whether that is necessary to accomplish the intended purposes for peering the VPCs.

Via CLI:

1. List all the route tables from a VPC and check if "GatewayId" is pointing to a *<peering_connection_id>* (e.g. pcx-1a2b3c4d) and if "DestinationCidrBlock" is as specific as desired.

```
aws ec2 describe-route-tables --filter "Name=vpc-id,Values=<vpc_id>" --query
"RouteTables[*].{RouteTableId:RouteTableId, VpcId:VpcId, Routes:Routes,
AssociatedSubnets:Associations[*].SubnetId}"
```

Remediation:

Remove and add route table entries to ensure that the least number of subnets or hosts as is required to accomplish the purpose for peering are routable.

Via CLI:

1. For each *<route_table_id>* containing routes non compliant with your routing policy (which grants more than desired "least access"), delete the non compliant route:

```
aws ec2 delete-route --route-table-id <route_table_id> --destination-cidr-block
<non_compliant_destination_CIDR>
```

2. Create a new compliant route:

```
aws ec2 create-route --route-table-id <route_table_id> --destination-cidr-block
<compliant_destination_CIDR> --vpc-peering-connection-id <peering_connection_id>
```

References:

1. http://docs.aws.amazon.com/AmazonVPC/latest/PeeringGuide/peering-configurations-partial-access.html

Appendix: Summary Table

Control		Set Correctly	
		Yes	No
1	**Identity and Access Management**		
1.1	Avoid the use of the "root" account (Scored)	☐	☐
1.2	Ensure multi-factor authentication (MFA) is enabled for all IAM users that have a console 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 MFA is enabled for the "root" account (Scored)	☐	☐
1.14	Ensure hardware MFA is enabled for the "root" account (Scored)	☐	☐
1.15	Ensure security questions are registered in the AWS account (Not Scored)	☐	☐
1.16	Ensure IAM policies are attached only to groups or roles (Scored)	☐	☐
1.17	Enable detailed billing (Scored)	☐	☐
1.18	Ensure IAM Master and IAM Manager roles are active (Scored)	☐	☐
1.19	Maintain current contact details (Scored)	☐	☐
1.20	Ensure security contact information is registered (Scored)	☐	☐
1.21	Ensure IAM instance roles are used for AWS resource access from instances (Not Scored)	☐	☐
1.22	Ensure a support role has been created to manage incidents with AWS Support (Scored)	☐	☐

1.23	Do not setup access keys during initial user setup for all IAM users that have a console password (Not Scored)	☐	☐
1.24	Ensure IAM policies that allow full "*:*" administrative privileges are not created (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 on the CloudTrail S3 bucket (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 authentication 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 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 VPCs (Scored)	☐	☐
4.4	Ensure the default security group of every VPC restricts all traffic (Scored)	☐	☐
4.5	Ensure routing tables for VPC peering are "least access" (Not Scored)	☐	☐

Appendix: Change History

Date	Version	Changes for this version
02-01-2016	1.0.0	Initial Release
11-09-2016	1.1.0	Added recommendation 1.3 (Ticket #69)
11-09-2016	1.1.0	Updated section 3 remediations (Ticket #88)
11-09-2016	1.1.0	1.4 - Updated commands (Ticket #59)
11-09-2016	1.1.0	1.14 - Updated commands (Ticket #61)
11-09-2016	1.1.0	1.3 - Updated commands (Ticket #58)
11-09-2016	1.1.0	1.13 - Updated commands (Ticket #60)
11-09-2016	1.1.0	4.4 - Updated Remediation (Ticket #67, #68)
11-09-2016	1.1.0	2.1 - Updated remediation (Ticket #66)
11-09-2016	1.1.0	1.14 - Updated Title (Ticket #73)
11-09-2016	1.1.0	3.15 - Moved to section 1 (Ticket #93)
11-09-2016	1.1.0	4.3 - Added note (Ticket #106)
11-09-2016	1.1.0	2.7 - Fixed typo in description (Ticket #54)
11-09-2016	1.1.0	3.1 - Added note to remediation (Ticket #109)
11-09-2016	1.1.0	Added Recommendation 1.24 (Ticket

		#103)
11-09-2016	1.1.0	Standardized benchmark example variables (Ticket #65)
11-09-2016	1.1.0	1.14 - Updated audit (Ticket #76)
11-09-2016	1.1.0	4.1, 4.2, 4.5 - Added warning (Ticket #87)
11-09-2016	1.1.0	3.2 - Updated audit (Ticket #75, #108)
11-09-2016	1.1.0	1.11 - Updated remediation (Ticket #53)
11-09-2016	1.1.0	2.2 - Fixed typo in description (Ticket #57)
11-09-2016	1.1.0	1.1 - Fixed typo in description and rationale (Ticket #56)
11-09-2016	1.1.0	2.1, 2.5, 2.6, 4.3 - Added notes (Ticket #89)