CIS Snowflake Foundations Benchmark

v1.0.0 - 10-27-2023

Terms of Use

Table of Contents

Overview

All CIS Benchmarks focus on technical configuration settings used to maintain and/or increase the security of the addressed technology, and they should be used in **conjunction** with other essential cyber hygiene tasks like:

- Monitoring the base operating system for vulnerabilities and quickly updating with the latest security patches
- Monitoring applications and libraries for vulnerabilities and quickly updating with the latest security patches

In the end, the CIS Benchmarks are designed as a key **component** of a comprehensive cybersecurity program.

This security configuration benchmark provides prescriptive guidance for configuring security options for Snowflake.
The recommendations detailed here are important security considerations when designing your workloads and applications on Snowflake. Most of the recommendations provided with this release of the benchmark cover security considerations only at individual account level and not at the organization level.

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 Snowflake.

Consensus Guidance

This CIS Benchmark was created using a consensus review process comprised of a global community of subject matter experts. The process combines real world experience with data-based information to create technology specific guidance to assist users to secure their environments. Consensus participants provide perspective from a diverse set of backgrounds including consulting, software development, audit and compliance, security research, operations, government, and legal.

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

Typographical Conventions

The following typographical conventions are used throughout this guide:

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

Recommendation Definitions

The following defines the various components included in a CIS recommendation as applicable. If any of the components are not applicable it will be noted or the component will not be included in the recommendation.

Title

Concise description for the recommendation's intended configuration.

Assessment Status

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

Automated

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

Manual

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

Profile

A collection of recommendations for securing a technology or a supporting platform. Most benchmarks include at least a Level 1 and Level 2 Profile. Level 2 extends Level 1 recommendations and is not a standalone profile. The Profile Definitions section in the benchmark provides the definitions as they pertain to the recommendations included for the technology.

Description

Detailed information pertaining to the setting with which the recommendation is concerned. In some cases, the description will include the recommended value.

Rationale Statement

Detailed reasoning for the recommendation to provide the user a clear and concise understanding on the importance of the recommendation.

Impact Statement

Any security, functionality, or operational consequences that can result from following the recommendation.

Audit Procedure

Systematic instructions for determining if the target system complies with the recommendation

Remediation Procedure

Systematic instructions for applying recommendations to the target system to bring it into compliance according to the recommendation.

Default Value

Default value for the given setting in this recommendation, if known. If not known, either not configured or not defined will be applied.

References

Additional documentation relative to the recommendation.

CIS Critical Security Controls® (CIS Controls®)

The mapping between a recommendation and the CIS Controls is organized by CIS Controls version, Safeguard, and Implementation Group (IG). The Benchmark in its entirety addresses the CIS Controls safeguards of (v7) “5.1 - Establish Secure Configurations” and (v8) '4.1 - Establish and Maintain a Secure Configuration Process” so individual recommendations will not be mapped to these safeguards.

Additional Information

Supplementary information that does not correspond to any other field but may be useful to the user.

Profile Definitions

The following configuration profiles are defined by this Benchmark:

- **Level 1**

  Items in this profile intend to:

  - be practical and prudent;
  - provide security focused best practice hardening of a technology; and
  - limit impact to 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 more critical than manageability and usability
  - acts as defense in depth measure
  - may impact the utility or performance of the technology
  - may include additional licensing, cost, or addition of third party software

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:

Authors
Iulia Ion
Oleg Mitrofanov
Cameron Tekiyeh

Contributors
Tim Harrison CISSP, ICP, KMP, Center for Internet Security, New York
Frantisek Rolinek
Anoosh Saboori
Prasoon Shukla
David Schultz
Shad Sharma
Iulia Ion
Phil White , Center for Internet Security, New York

Editor
Phil White , Center for Internet Security, New York

Recommendations

1 Identity and Access Management

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

1.1 Ensure single sign-on (SSO) is configured for your account / organization (Automated)

Profile Applicability:

- Level 1

Description:

Federated authentication enables users to connect to Snowflake using secure SSO (single sign-on). With SSO enabled, users authenticate through an external (SAML 2.0-compliant or OAuth 2.0) identity provider (IdP). Once authenticated by an IdP, users can access their Snowflake account for the duration of their IdP session without having to authenticate to Snowflake again. Users can choose to initiate their sessions from within the interface provided by the IdP or directly in Snowflake.

Snowflake offers native support for federated authentication and SSO through Okta and Microsoft ADFS.

Snowflake also supports most SAML 2.0-compliant vendors as an IdP, including Google G Suite, Microsoft Azure Active Directory, OneLogin, and Ping Identity PingOne. To use an IdP other than Okta or ADFS, you must define a custom application for Snowflake in the IdP.

There are two ways to configure SAML:

- By creating the security integration (recommended)
- By setting the `SAML_IDENTITY_PROVIDER` account parameter (deprecated)

Rationale:

Configuring your Snowflake authentication so that users can log in using SSO reduces the attack surface for your organization because users only log in once across multiple applications and do not have to manage a separate set of credentials for their Snowflake account.

Impact:

There may be costs associated with provisioning and using an IdP service.

Audit:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. List all the security integrations in your account.

```
SHOW SECURITY INTEGRATIONS;
```

2. Ensure that there are security integrations of type `SAML2` and `EXTERNAL_OAUTH` configured for an account.

```
SELECT *
FROM TABLE(RESULT_SCAN(LAST_QUERY_ID()))
WHERE ("type" LIKE 'EXTERNAL_OAUTH%' OR "type" LIKE 'SAML2')
  AND "enabled" = TRUE;
```

3. Ensure that there is an SSO integration configured for the account.
Note: The presence of a configured security integration does not mean that it is configured correctly and working. Configuration correctness should be explicitly tested

Required privileges:
To be able to execute the above audit query above, the caller needs the `USAGE` privilege on every security integration in your Snowflake account.

Remediation:

The steps for configuring an IdP differ depending on whether you choose SAML2 or OAuth. They further differ depending on what identity provider you choose: Okta, AD FS, Ping Identity, Azure AD, or custom. For specific instructions, see Snowflake documentation on SAML and External OAuth.
Note: If your SAML integration is configured using the deprecated account parameter `SAML_IDENTITY_PROVIDER`, you should migrate to creating a security integration using the `system$migrate_saml_idp_registration` function. For more information, see the Migrating to a SAML2 Security Integration documentation.

Default Value:

By default, Snowflake is not configured to use SSO-based authentication.

References:

1. https://docs.snowflake.com/en/user-guide/admin-security-fed-auth.html
2. https://docs.snowflake.com/en/user-guide/admin-security-fed-auth-configure-idp.html
3. https://docs.snowflake.com/en/user-guide/oauth-external.html
4. https://docs.snowflake.com/en/user-guide/admin-security-fed-auth-advanced.html
5. https://docs.snowflake.com/en/sql-reference/parameters#saml-identity-provider
6. https://docs.snowflake.com/en/user-guide/admin-security-fed-auth-configure-snowflake

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.6 Centralize Account Management Centralize account management through a directory or identity service.		●	●
v7	16.2 Configure Centralized Point of Authentication Configure access for all accounts through as few centralized points of authentication as possible, including network, security, and cloud systems.		●	●

1.2 Ensure Snowflake SCIM integration is configured to automatically provision and deprovision users and groups (i.e. roles) (Automated)

Profile Applicability:

- Level 2

Description:

The System for Cross-domain Identity Management (SCIM) is an open specification designed to help facilitate the automated management of user identities and groups (i.e. roles) in cloud applications using RESTful APIs.

Snowflake supports SCIM 2.0 integration with Okta, Microsoft Azure AD and custom identity providers. Users and groups from the identity provider can be provisioned into Snowflake, which functions as the service provider.

Rationale:

While SSO enables seamless authentication with a federated identity to the Snowflake application, user accounts still need to be created, managed, and deprovisioned. Operations like adding and deleting users, changing permissions, and adding new types of accounts usually take up valuable admin time and when done manually may be error-prone.

With SCIM, user identities can be created either directly in your identity provider, or imported from external systems like HR software or Active Directory. SCIM enables IT departments to automate the user provisioning and deprovisioning process while also having a single system to manage permissions and groups. Since data is transferred automatically, risk of error is reduced.

Impact:

None.

Audit:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. List the security integrations in your account:

```
SHOW SECURITY INTEGRATIONS;
```

2. Ensure that a SCIM integration is configured. The output of the following command shows all SCIM security integrations configured for an account. No output means no SCIM integration has been configured for the account.
3. Ensure that there are security integrations of type `SCIM%` with `enabled` set to `true`.

```
SELECT *
FROM TABLE(result_scan(last_query_id()))
WHERE ("type" like 'SCIM%') AND "enabled" = true;
```

Note: The presence of a SCIM security integration does not mean that it is configured correctly and working.

Required privileges:
To be able to execute the above audit query above, the caller needs the `USAGE` privilege on every security integration in an account.

Remediation:

Follow the instructions in the Snowflake documentation to set up SCIM configuration for Okta, Azure AD, or configure a custom SCIM integration.

Default Value:

By default, SCIM integration is not set-up for Snowflake.

References:

1. https://docs.snowflake.com/en/user-guide/scim-intro.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.6 Centralize Account Management Centralize account management through a directory or identity service.		●	●
v8	6.8 Define and Maintain Role-Based Access Control Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●
v7	16.2 Configure Centralized Point of Authentication Configure access for all accounts through as few centralized points of authentication as possible, including network, security, and cloud systems.		●	●

1.3 Ensure that Snowflake password is unset for SSO users (Manual)

Profile Applicability:

- Level 1

Description:

Ensure that Snowflake password is unset for SSO users.

Rationale:

Allowing users to sign in with Snowflake passwords in the presence of a configured third-party identity provider SSO may undermine mandatory security controls configured on the SSO and degrade the security posture of the account. For example, the SSO sign-in flow may be configured to require multi-factor authentication (MFA), whereas the Snowflake password sign-in flow may not.

Note:

- This benchmark does not preclude configuration of key pair authentication for SSO users. Key pair authentication may be necessary for users to interact with Snowflake programmatically or through third party tools.
- To mitigate the risk of users not being able to sign-in due to SSO provider outage, ensure that at least one SSO break-glass user exists with Snowflake password reset privileges for account users. This break-glass user should be able to sign in using a Snowflake native password (coupled with MFA) or a key pair.

Impact:

Users will not be able to sign into their Snowflake accounts if SSO sign-in flow breaks, for example due to SSO provider outage.

Audit:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. Show all users in an account that have a password set:

```
SELECT NAME, HAS_PASSWORD
FROM SNOWFLAKE.ACCOUNT_USAGE.USERS
WHERE HAS_PASSWORD
  AND DELETED_ON IS NULL
  AND NOT DISABLED;
```

2. Check your IdP configurations and ensure that, if there are users with passwords, these users are not SSO users. An exception should be allowed for a the break-glass SSO user which needs to be able to log-in with a Snowflake password and MFA (or with a key pair).

Required privileges:
Running the query requires the `SECURITY_VIEWER` role on the `Snowflake` database.

Remediation:

Programmatically:
For each SSO user `<username>` with a password, run the following command to set password to `null`:

```
ALTER USER <username>
     SET PASSWORD = NULL;
```

Default Value:

When a user is created using the `CREATE USER` command in Snowflake, providing a password is optional. When a password is not specified, it is set to `NULL`.

When System for Cross-domain Identity Management (SCIM) integration is configured and users are managed in an external identity provider, whether a Snowflake password is set for a user by default depends on the default configuration of the SCIM client. For example, the Okta SCIM client by default is configured to generate and set a new random password whenever the user's Okta password changes.

References:

1. https://docs.snowflake.com/en/sql-reference/sql/create-user.html
2. https://docs.snowflake.com/en/user-guide/scim-okta.html#features
3. https://docs.snowflake.com/en/user-guide/key-pair-auth.html
4. https://community.snowflake.com/s/article/FAQ-User-and-Password-Management

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	0.0 Explicitly Not Mapped Explicitly Not Mapped			
v7	0.0 Explicitly Not Mapped Explicitly Not Mapped			

1.4 Ensure multi-factor authentication (MFA) is turned on for all human users with password-based authentication (Automated)

Profile Applicability:

- Level 1

Description:

Multi-factor authentication (MFA) is a security control used to add an additional layer of login security. It works by requiring the user to present two or more proofs (factors) of user identity. An MFA example would be requiring a password and a verification code delivered to the user's phone during user sign-in.

The MFA feature for Snowflake users is powered by the Duo Security service.

Rationale:

MFA mitigates security threats of users creating weak passwords and user passwords being stolen or accidentally leaked.

Impact:

If users lose access to the second factor of authentication, an account admin may need to reset their access.

Audit:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. List all active users with passwords and no MFA enabled:

```
SELECT NAME, EXT_AUTHN_DUO AS MFA_ENABLED
FROM SNOWFLAKE.ACCOUNT_USAGE.USERS
WHERE DELETED_ON IS NULL
  AND NOT DISABLED
  AND HAS_PASSWORD;
```

2. Ensure that the query above does not return any results.
   Note: If users have SSO enabled, the MFA authentication will be handled by the Identity Provider and does not reflect in the query above. For SSO users, configure and check MFA status on your Identity Provider.

Required privileges:
Running the query requires the `SECURITY_VIEWER` role on the `Snowflake` database.

Remediation:

Users have to individually enroll into MFA using the Snowflake web UI.
From the UI:

1. Each user with a password should go to https://app.snowflake.com/ and sign into their Snowflake account.
2. Click on the username on the top left side.
3. Click on `Profile`.
4. Next to `Multi-factor authentication` click `Enroll`.
5. Click `Start setup`.
6. Select the type of device and click `Continue`.
7. Follow the steps to finish the enrollment.

If MFA needs to be enabled for a large population of users, consider prioritizing users with `ACCOUNTADMIN`, `SECURITYADMIN` or other highly privileged roles.
For specific instructions, see the documentation page Enrolling in MFA (Multi-Factor Authentication).
Note: If you use SSO authentication, you will have to check and configure MFA with your Identity Provider.

Default Value:

By default MFA is not enabled for Snowflake users.

References:

1. https://docs.snowflake.com/en/user-guide/security-access-control-configure.html
2. https://docs.snowflake.com/en/user-guide/security-mfa.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.3 Require MFA for Externally-Exposed Applications Require all externally-exposed enterprise or third-party applications to enforce MFA, where supported. Enforcing MFA through a directory service or SSO provider is a satisfactory implementation of this Safeguard.		●	●
v8	6.5 Require MFA for Administrative Access Require MFA for all administrative access accounts, where supported, on all enterprise assets, whether managed on-site or through a third-party provider.	●	●	●
v7	4.5 Use Multifactor Authentication For All Administrative Access Use multi-factor authentication and encrypted channels for all administrative account access.		●	●
v7	16.3 Require Multi-factor Authentication Require multi-factor authentication for all user accounts, on all systems, whether managed onsite or by a third-party provider.		●	●

1.5 Ensure minimum password length is set to 14 characters or more (Automated)

Profile Applicability:

- Level 1

Description:

To mitigate the risk of unauthorized access to a Snowflake account through easily guessable password, Snowflake enforces the following password policy as a minimum requirement while using the `ALTER USER` command and the web interface:

- Must be at least 8 characters long.
- Must contain at least 1 digit.
- Must contain at least 1 uppercase letter and 1 lowercase letter.

Snowflake password policies can be used to specify and enforce further constraints on password length and complexity.

Snowflake supports setting a password policy for your Snowflake account and for individual users. Only one password policy can be set at any given time for your Snowflake account or a user. If a password policy exists for the Snowflake account and another password policy is set for a user in the same Snowflake account, the user-level password policy takes precedence over the account-level password policy.

The password policy applies to new passwords that are set in your Snowflake account. To ensure that users with existing passwords meet the password policy requirements, require users to change their password during their next login to Snowflake as shown in Step 6: Require a Password Change.

Rationale:

While Snowflake recommends configuring SSO authentication for users and ensuring that SSO users do not have a password set, there may be exceptions when users still need to log in with a password (e.g., setting up a break-glass user with password login to recover from SSO outages). For those few users that still need to have a password, setting a password policy can help ensure that, throughout subsequent password changes, the passwords used remain complex and therefore harder to guess or brute-force.

Impact:

None.

Audit:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. List account-level password policies that enforce a minimum password length of 14 characters.

```
WITH PWDS_WITH_MIN_LEN AS (
         SELECT ID
  FROM SNOWFLAKE.ACCOUNT_USAGE.PASSWORD_POLICIES
  WHERE PASSWORD_MIN_LENGTH >= 14
         AND DELETED IS NULL
)
SELECT A.*
FROM SNOWFLAKE.ACCOUNT_USAGE.POLICY_REFERENCES AS A
  LEFT JOIN PWDS_WITH_MIN_LEN AS B ON A.POLICY_ID = B.ID
WHERE A.REF_ENTITY_DOMAIN = 'ACCOUNT'
  AND A.POLICY_KIND = 'PASSWORD_POLICY'
  AND A.POLICY_STATUS = 'ACTIVE'
  AND B.ID IS NOT NULL;
```

2. Ensure that the query above returns a password policy.
3. List all user-level password policies. All password policies applied on the user level also need to be checked, therefore a password policy set for a user overrides a password policy set on an account.

```
WITH PWDS_WITH_MIN_LEN AS (
  SELECT ID
  FROM SNOWFLAKE.ACCOUNT_USAGE.PASSWORD_POLICIES
  WHERE PASSWORD_MIN_LENGTH >= 14
         AND DELETED IS NULL
)
SELECT A.*
FROM SNOWFLAKE.ACCOUNT_USAGE.POLICY_REFERENCES AS A
  LEFT JOIN PWDS_WITH_MIN_LEN AS B ON A.POLICY_ID = B.ID
WHERE A.REF_ENTITY_DOMAIN = 'USER'
  AND A.POLICY_STATUS = 'ACTIVE'
  AND B.ID IS NULL;
```

4. Ensure that the query above does not return any results.

Remediation:

Follow the following steps to set and enforce a password policy:

1. Create the password policy if it does not exist:

```
CREATE PASSWORD POLICY <password_policy>
  PASSWORD_MIN_LENGTH = 14
  PASSWORD_MAX_AGE_DAYS = 0;
```

2. Set password policy on the account level:

```
ALTER ACCOUNT
  SET PASSWORD POLICY <password_policy>;
```

Note: It may take up to 2 hours for the password policies created to show up in the account usage view. For more information on latency, see the Data latency for Account Usage documentation.

Default Value:

The default value for minimum password length is 8 characters.

References:

1. https://docs.snowflake.com/en/user-guide/admin-user-management#password-policies

Additional Information:

Snowflake password policies are effective only for Snowflake users authenticating with Snowflake-managed passwords. For users accessing Snowflake accounts integrated with an SSO provider, password policy should be set on the SSO provider side.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.2 Use Unique Passwords Use unique passwords for all enterprise assets. Best practice implementation includes, at a minimum, an 8-character password for accounts using MFA and a 14-character password for accounts not using MFA.	●	●	●
v7	4.4 Use Unique Passwords Where multi-factor authentication is not supported (such as local administrator, root, or service accounts), accounts will use passwords that are unique to that system.		●	●

1.6 Ensure that service accounts use key pair authentication (Automated)

Profile Applicability:

- Level 1

Description:

Service account is an identity used by scripts, jobs, applications, pipelines, etc. to talk to Snowflake. It is also sometimes known as "application user", "service principal", "system account", or "daemon user".

On the platform level Snowflake does not differentiate between Snowflake users created for and used by humans and Snowflake users created for and used by services.

Password-based authentication used by humans can be augmented by a second factor (MFA), e.g. a hardware token, or a security code pushed to a mobile device. Services and automation cannot be easily configured to authenticate with a second factor. Instead, for such use cases, Snowflake supports using key pair authentication as a more secure alternative to password-based authentication.

Note that password-based authentication for a service account can be enabled along with a key-based authentication. To ensure that only key-based authentication is enabled for a service account, the `PASSWORD` parameter for that Snowflake user must be set to `null`.

Rationale:

Password-based authentication has a set of disadvantages that increase probability of a security incident, especially when used without MFA:

- Passwords created by humans are generally more predictable and less random than keys generated by a computer. Consequently, passwords are easier to brute force both online (against a live service) or offline (against a hashed password database).
- Passwords are usually transmitted over the network and can be leaked when the transmission channel is insecure or when an application is accidentally misconfigured to log passwords.
- Passwords are easier to leak by writing them down on a sticky note attached to the back of a keyboard.
- It is easier to trick (phish) a user into revealing their password to an unauthorized party.

Using key-based authentication for service accounts helps with mitigating the aforementioned issues.

Impact:

Snowflake authentication for existing automation and services that use service accounts with password-based authentication will be broken if corresponding configuration is not updated before service accounts passwords are set to null.

Audit:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. If Snowflake service account users are marked with `ACCOUNT_TYPE=service` tag, then all non-compliant service account users that either have a password or do not have key authentication enabled can be identified with the following query:

```
-- The query assumes that service accounts are tagged with
ACCOUNT_TYPE=service tag.
select tr.object_name
from snowflake.account_usage.tag_references tr
left join snowflake.account_usage.users u on tr.object_name = u.name
where  tr.tag_name = 'ACCOUNT_TYPE'
         and tr.tag_value = 'service'
         and tr.domain = 'USER'
         and u.deleted_on is null
  and (u.has_password = true OR has_rsa_public_key = false);
```

2. Ensure that the query above does not return any results.

Required privileges:
The query requires the following privileges:

- Database role `snowflake.security_viewer`.
- Database role `snowflake.governance_viewer`.

Remediation:

Programmatically:
For every non-compliant service account:

1. Follow the Configuring Key Pair Authentication instructions to generate the key <rsa_public_key>.
2. In a Snowsight worksheet or through the SnowSQL CLI, run the following command:

```
ALTER USER <service_account_name>
  SET RSA_PUBLIC_KEY='<rsa_public_key>';
```

3. Update configuration of the automation and services that rely on the service account to use key-based authentication. This is going to be specific to the service in question.
4. Disable password-based authentication:

```
ALTER USER <service_account_name> SET PASSWORD = null;
```

Default Value:

To enable key based authentication for a Snowflake user either `RSA_PUBLIC_KEY` or `RSA_PUBLIC_KEY_2` parameters must be set on the Snowflake user. By default the CREATE USER command does not require setting either of the parameters. Also, setting either of the parameters does not prevent from additionally setting a password for the same user.

References:

1. https://docs.snowflake.com/en/user-guide/key-pair-auth.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	0.0 Explicitly Not Mapped Explicitly Not Mapped			
v7	0.0 Explicitly Not Mapped Explicitly Not Mapped			

1.7 Ensure authentication key pairs are rotated every 180 days (Automated)

Profile Applicability:

- Level 1

Description:

Snowflake supports using RSA key pair authentication as an alternative to password authentication and as a primary way to authenticate service accounts.

Authentication key pair rotation is a process of replacing an existing authentication key pair with a freshly generated key pair.

Snowflake supports two active authentication key pairs to allow for uninterrupted key rotation. Rotate and replace your authentication key pairs based on the expiration schedule at least once every 180 days.

Rationale:

Periodic authentication key pair rotation mitigates the threat of compromised or leaked keys. It reduces the window of opportunity during which a given key is valid and can be used by a threat actor.

Impact:

Existing automation and services that rely on key pair authentication may break if they are not updated to use a new authentication key before the old key is inactivated.

Audit:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. Parse the account query history and output all users and key pair names for key pairs that were set more than 180 days ago.

```
WITH FILTERED_QUERY_HISTORY AS (
    -- Extract necessary fields and apply initial filters
    SELECT END_TIME AS SET_TIME,
        UPPER(REGEXP_SUBSTR(QUERY_TEXT, 'USER\\s+"?([\\w]+)"?', 1, 1,
'i', 1)) AS PROCESSED_USERNAME,
        QUERY_TEXT
    FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY
    WHERE EXECUTION_STATUS = 'SUCCESS'
        AND QUERY_TYPE IN ('ALTER_USER', 'CREATE_USER')
        AND TO_DATE(SET_TIME) < DATEADD(day, -180, CURRENT_DATE())
        AND (QUERY_TEXT ILIKE '%rsa_public_key%' OR QUERY_TEXT ILIKE
'%rsa_public_key_2%')
),
EXTRACTED_KEYS AS (
    SELECT SET_TIME,
        PROCESSED_USERNAME,
        CASE
            WHEN POSITION('rsa_public_key' IN LOWER(QUERY_TEXT)) > 0
THEN 'rsa_public_key'
            WHEN POSITION('rsa_public_key_2' IN LOWER(QUERY_TEXT)) > 0
THEN 'rsa_public_key_2'
            ELSE NULL
        END AS RSA_KEY_NAME
    FROM FILTERED_QUERY_HISTORY
    WHERE POSITION('rsa_public_key' IN LOWER(QUERY_TEXT)) > 0 OR
POSITION('rsa_public_key_2' IN LOWER(QUERY_TEXT)) > 0
),
RECENT_KEYS AS (
    SELECT EK.SET_TIME,
        EK.PROCESSED_USERNAME AS USERNAME,
        EK.RSA_KEY_NAME AS RSA_PUBLIC_KEY,
        ROW_NUMBER() OVER (PARTITION BY ek.processed_username,
ek.rsa_key_name ORDER BY ek.set_time DESC) AS rnum
    FROM EXTRACTED_KEYS EK
        INNER JOIN SNOWFLAKE.ACCOUNT_USAGE.USERS AU ON
EK.PROCESSED_USERNAME = AU.NAME
    WHERE AU.DELETED_ON IS NULL
        AND AU.DISABLED = FALSE
        AND EK.RSA_KEY_NAME IS NOT NULL
)
-- Select the most recent RSA key name for each user
SELECT SET_TIME,
    USERNAME,
    RSA_PUBLIC_KEY
FROM RECENT_KEYS
WHERE RNUM = 1;
```

2. Ensure that the query above does not return any results.

Note: This query above is limited by the query history length that goes back to 360 days only. Key pairs set more than 360 days ago will not be surfaced by this query.
Required privileges:
To run the query above, the caller needs the:

- The `SECURITY_VIEWER` role on the `Snowflake` database.
- The `GOVERNANCE_VIEWER` role on the `Snowflake` database.

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. For every Snowflake service account whose authentication key pair age is >= 180 days, generate a new RSA authentication keypair.
2. Update either `RSA_PUBLIC_KEY` and `RSA_PUBLIC_KEY_2` properties of a user, whichever is currently unset.

```
ALTER USER <username> SET RSA_PUBLIC_KEY_2='JERUEHtcve...';
```

3. Identify all services and automation that authenticate using existing keypair and update them to authenticate using freshly generated keypair.
4. Unset either `RSA_PUBLIC_KEY` or `RSA_PUBLIC_KEY_2` properties of a user, whichever is assigned the old public key.

```
ALTER USER <username> UNSET RSA_PUBLIC_KEY;
```

For more information, see Configuring Key Pair Rotation.

Default Value:

No authentication key pairs are rotated automatically.

References:

1. https://docs.snowflake.com/en/user-guide/key-pair-auth.html#configuring-key-pair-rotation

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	0.0 Explicitly Not Mapped Explicitly Not Mapped			
v7	0.0 Explicitly Not Mapped Explicitly Not Mapped			

1.8 Ensure that users who did not log in for 90 days are disabled (Automated)

Profile Applicability:

- Level 1

Description:

Access grants tend to accumulate over time unless explicitly set to expire. Regularly revoking unused access grants and disabling inactive user accounts is a good countermeasure to this dynamic.

If credentials of an inactive user account are leaked or stolen, it may take longer to discover the compromise.

In Snowflake an user account can be disabled by users with the `ACCOUNTADMIN` role.

Rationale:

Disabling inactive user accounts supports the principle of least privilege and generally reduces attack surface.

Impact:

There is a chance of disabling users or service accounts that are used consistently, but very infrequently, e.g. once or twice a year. Such users should be tagged and filtered out in the audit query.

Audit:

From the UI:

1. Go to https://app.snowflake.com/ and sign into your Snowflake account.
2. On the left side navigation bar, click on *Admin*.
3. Under *Admin*, click on *Users & Roles*.
4. Under the *Users* tab, sort by `LAST LOGIN`.
5. Ensure that all users whose last login is older than 90 days have `STATUS` set to `Disabled`.

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. List all users in the account

```
SHOW USERS;
```

2. For each user, ensure that if `disabled` is set to `false`, the value of the `last_success_login` field is less than 90 days ago.

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:
For each user `<user_name>` that has not logged in in the last 90 days, run the following query to disable their account:

```
ALTER USER  <user_name> SET DISABLED = true;
```

If there is a need for re-enabling an account, a user must contact one of the Snowflake account administrative users.

Default Value:

By default Snowflake users are not disabled due to inactivity. An `ACCOUNTADMIN` must explicitly disable an inactive user.

References:

1. https://docs.snowflake.com/en/user-guide/admin-user-management.html#disabling-enabling-a-user

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.3 Disable Dormant Accounts Delete or disable any dormant accounts after a period of 45 days of inactivity, where supported.	●	●	●
v7	16.9 Disable Dormant Accounts Automatically disable dormant accounts after a set period of inactivity.	●	●	●

1.9 Ensure that the idle session timeout is set to 15 minutes or less for users with the ACCOUNTADMIN and SECURITYADMIN roles (Automated)

Profile Applicability:

- Level 1

Description:

A session begins when a user connects to Snowflake and authenticates successfully using a Snowflake programmatic client, Snowsight, or the classic web interface.

A session is maintained indefinitely with continued user activity. After a period of inactivity in the session, known as the idle session timeout, the user must authenticate to Snowflake again. Session policies can be used to modify the idle session timeout period. The idle session timeout has a maximum value of four hours.

Rationale:

Tightening up the idle session timeout reduces sensitive data exposure risk when users forget to sign out of Snowflake and an unauthorized person gains access to their device.

Impact:

Too short idle session timeout may result in poor user experience due to users continuously being logged out of their accounts.

Audit:

Programmatically:
In the Snowsight UI or from the SnowSQL CLI:

1. Identify all users with the `ACCOUNTADMIN` and `SECURITYADMIN` roles with session timeout greater than 15 minutes or not explicitly set, thus defaulting to 240 minutes.

```
--SESSION POLICIES APPLIED TO PRIVILEGE USERS DIRECTLY
--FIND ALL PRIVILEGED USERS
WITH PRIV_USERS AS (
  SELECT DISTINCT GRANTEE_NAME
  FROM SNOWFLAKE.ACCOUNT_USAGE.GRANTS_TO_USERS
WHERE DELETED_ON IS NULL
  AND ROLE IN ('ACCOUNTADMIN','SECURITYADMIN')
  AND DELETED_ON IS NULL
)
--CHECK IF THERE IS AN ACTIVE SESSION POLICY OF 15 MINUTES CREATED FOR
USERS
, POLICY_REFS AS (
SELECT *
FROM SNOWFLAKE.ACCOUNT_USAGE.POLICY_REFERENCES AS A
LEFT JOIN SNOWFLAKE.ACCOUNT_USAGE.SESSION_POLICIES AS B ON A.POLICY_ID
= B.ID
WHERE A.POLICY_KIND = 'SESSION_POLICY'
  AND A.POLICY_STATUS = 'ACTIVE'
  AND A.REF_ENTITY_DOMAIN = 'USER'
  AND B.DELETED IS NULL
  AND B.SESSION_IDLE_TIMEOUT_MINS <= 15
)
--SHOW ALL PRIVILEGED USERS THAT DO NOT HAVE THE SESSION POLICY APPLIED
SELECT A.*,
  B.POLICY_ID,
  B.POLICY_KIND,
  B.POLICY_STATUS,
  B.SESSION_IDLE_TIMEOUT_MINS
FROM PRIV_USERS AS A
LEFT JOIN POLICY_REFS AS B ON A.GRANTEE_NAME = B.REF_ENTITY_NAME
WHERE B.POLICY_ID IS NULL;
```

2. Ensure that the query above does not return any users.
3. In addition to user-attached session policies, session policies can be applied at an account level. The following query will check if there is a satisfactory session policy created at an account level, which would by default be applied to all users (including privileged users).

```
--SESSION POLICIES APPLIED TO AT AN ACCOUNT LEVEL
SELECT *
FROM SNOWFLAKE.ACCOUNT_USAGE.POLICY_REFERENCES AS A
LEFT JOIN SNOWFLAKE.ACCOUNT_USAGE.SESSION_POLICIES AS B ON A.POLICY_ID
= B.ID
WHERE A.POLICY_KIND = 'SESSION_POLICY'
  AND A.POLICY_STATUS = 'ACTIVE'
  AND A.REF_ENTITY_DOMAIN = 'ACCOUNT'
  AND B.DELETED IS NULL
  AND B.SESSION_IDLE_TIMEOUT_MINS <= 15;
```

4. Ensure that the query above returns a result.
   Note: Latency for the session policy view can be up to 2 hours.

Remediation:

Programmatically:
In the Snowsight UI or from the SnowSQL CLI:

1. Create the session policy if it does not exist yet. Execute the following commands to create and set the idle session timeout for highly privileged users in your Snowflake account:

```
CREATE SESSION POLICY <session_policy>
  SESSION_IDLE_TIMEOUT_MINS = 15,
  SESSION_UI_IDLE_TIMEOUT_MINS = 15;
```

2. Set session policy for every highly privileged user.

```
ALTER USER <username> SET SESSION POLICY <session_policy>;
```

Default Value:

The default value for Snowflake idle session timeout is 4 hours.

References:

1. https://docs.snowflake.com/en/user-guide/session-policies
2. https://docs.snowflake.com/en/user-guide/session-policies#step-3-create-a-new-session-policy

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.3 Configure Automatic Session Locking on Enterprise Assets Configure automatic session locking on enterprise assets after a defined period of inactivity. For general purpose operating systems, the period must not exceed 15 minutes. For mobile end-user devices, the period must not exceed 2 minutes.	●	●	●
v7	16.11 Lock Workstation Sessions After Inactivity Automatically lock workstation sessions after a standard period of inactivity.	●	●	●

1.10 Limit the number of users with ACCOUNTADMIN and SECURITYADMIN (Automated)

Profile Applicability:

- Level 1

Description:

By default, `ACCOUNTADMIN` is the most powerful role in a Snowflake account. Users with the `SECURITYADMIN` role grant can trivially escalate their privileges to that of `ACCOUNTADMIN`.

Following the principle of least privilege that prescribes limiting user's privileges to those that are strictly required to do their jobs, the `ACCOUNTADMIN` and `SECURITYADMIN` roles should be assigned to a limited number of designated users (e.g., less than 10, but at least 2 to ensure that access can be recovered if one `ACCOUNTAMIN` user is having login difficulties).

Rationale:

While it is important to apply the principle of least privilege to all access grants, it is especially important to apply it to highly privileged roles. Examples of such roles are `ACCOUNTADMIN`, `SECURITYADMIN` and their equivalents. The fewer users with full administrator privileges, the smaller the attack surface and the probability of a full account compromise.

Impact:

Users who lose the `ACCOUNTADMIN` or `SECURITYADMIN` role grant and are not granted a more scoped down role appropriate to their job function may lose certain privileges required to do their job.

Audit:

From the UI:

1. Go to https://app.snowflake.com/ and sign into your Snowflake account.
2. On the left side navigation bar, click on *Admin*.
3. Under *Admin*, click on *Users & Roles*.
4. Ensure that a limited number of users have the `ACCOUNTADMIN` or `SECURITYADMIN` roles.

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. List all the users granted the `ACCOUNTADMIN` or `SECURITYADMIN` roles:

```
SELECT DISTINCT A.GRANTEE_NAME AS NAME,
  A.ROLE
FROM SNOWFLAKE.ACCOUNT_USAGE.GRANTS_TO_USERS AS A
  LEFT JOIN SNOWFLAKE.ACCOUNT_USAGE.USERS AS B ON A.GRANTEE_NAME =
B.NAME
WHERE A.ROLE IN ('ACCOUNTADMIN', 'SECURITYADMIN')
  AND A.DELETED_ON IS NULL
  AND B.DELETED_ON IS NULL
  AND NOT B.DISABLED
ORDER BY A.ROLE;
```

2. Ensure that the query above returns a small number of users (e.g., less than 10). To ensure that access is not being lost, the number of account users should be at least 2.

Privileges required:
To be able to execute the query above, the caller must have the role `SECURITY_VIEWER` on the `Snowflake` databases.

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. For each user `<username>` that does not need all the privileges a role provides to fulfill their job responsibilities, revoke the `ACCOUNTADMIN` or all equivalently privileged roles.

```
REVOKE ROLE ACCOUNTADMIN FROM USER <username>
```

2. For each user `<username>` that does not need all the privileges a role provides to fulfill their job responsibilities, revoke the `SECURITYADMIN` or all equivalently privileged roles.

```
REVOKE ROLE SECURITYADMIN FROM USER <username>
```

Default Value:

By default, only the user who creates a Snowflake account is assigned the `ACCOUNTADMIN` role.

References:

1. https://docs.snowflake.com/en/user-guide/security-access-control-considerations.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	0.0 Explicitly Not Mapped Explicitly Not Mapped			
v7	0.0 Explicitly Not Mapped Explicitly Not Mapped			

1.11 Ensure that all users granted the ACCOUNTADMIN role have an email address assigned (Automated)

Profile Applicability:

- Level 1

Description:

Every Snowflake user can be assigned an email address. The email addresses are then used by Snowflake features like notification integration, resource monitor and support cases to deliver email notifications to Snowflake users. In trial Snowflake accounts these email addresses are used for password reset functionality.

The email addresses assigned to `ACCOUNTADMIN` users are used by Snowflake to notify administrators about important events related to their accounts. For example, `ACCOUNTADMIN` users are notified about impending expiration of SAML2 certificates or SCIM access tokens.

Rationale:

If users with the `ACCOUNTADMIN` role are not assigned working email addresses that are being monitored and if SAML2 certificate used in SSO integration is not proactively renewed, expiration of SAML2 certificate may break the SSO authentication flow. Similarly, uncaught expiration of SCIM access token may break the SCIM integration.

Additionally, emails assigned to `ACCOUNTADMIN` users can be used by Snowflake Support to contact account administrators in urgent situations.

Impact:

None.

Audit:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. List users with no email address set:

```
SELECT DISTINCT a.grantee_name as name, b.email
FROM snowflake.account_usage.grants_to_users AS a
LEFT JOIN snowflake.account_usage.users AS b
    ON a.grantee_name = b.name
WHERE a.role = 'ACCOUNTADMIN'
  AND a.deleted_on IS NULL
  AND b.email IS NULL
  AND b.deleted_on IS NULL
  AND NOT b.disabled;
```

2. Ensure that the query above does not return any results.

Required privileges:
To be able to execute the above audit query, the caller must have the `SECURITY_VIEWER` role on the `SNOWFLAKE` database.

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. For every `ACCOUNTADMIN` user `<username>` that does not have email assigned run the following command to assign it:

```
ALTER USER <username>
  SET EMAIL = <email_address>;
```

Default Value:

The trial account creation form requires an email address. The first user of a newly created trial account is assigned that email address by default. All other users created in a Snowflake account must be assigned email addresses explicitly.

References:

1. https://docs.snowflake.com/en/user-guide/admin-user-management.html#resetting-the-password-for-an-administrator

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	0.0 Explicitly Not Mapped Explicitly Not Mapped			
v7	0.0 Explicitly Not Mapped Explicitly Not Mapped			

1.12 Ensure that no users have ACCOUNTADMIN or SECURITYADMIN as the default role (Automated)

Profile Applicability:

- Level 1

Description:

The ACCOUNTADMIN system role is the most powerful role in a Snowflake account and is intended for performing initial setup and managing account-level objects. `SECURITYADMIN` role can trivially escalate their privileges to that of `ACCOUNTADMIN`. Neither of these roles should be used for performing daily non-administrative tasks in a Snowflake account.

Instead, users should be assigned custom roles containing only those privileges that are necessary for successfully completing their job responsibilities.

Rationale:

When ACCOUNTADMIN is not set as a default user role, it forces account administrators to explicitly change their role to `ACCOUNTADMIN` each time they log in. This can help make account administrators aware of the purpose of roles in the system, prevent them from inadvertently using the `ACCOUNTADMIN` role for non-administrative tasks, and encourage them to change to the appropriate role for a given task. Same logic applies to the `SECURITYADMIN` role.

Impact:

None.

Audit:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. List users with `ACCOUNTADMIN` or `SECURITYADMIN` as the default role:

```
SELECT NAME, DEFAULT_ROLE
FROM SNOWFLAKE.ACCOUNT_USAGE.USERS
WHERE DEFAULT_ROLE IN ('ACCOUNTADMIN', 'SECURITYADMIN')
  AND DELETED_ON IS NULL
  AND NOT DISABLED;
```

2. Ensure that the query above does not return any users.

Required privileges:
Running the query above requires the `security_viewer` role on the `Snowflake` database

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. For each user `<user_name>` who has `ACCOUNTADMIN` or `SECURITYADMIN` as their default role, choose a less privileged role `<job_appropriate_role>` appropriate for their daily job responsibilities and run the following query:

```
ALTER USER <user_name>
  SET DEFAULT_ROLE = <job_appropriate_role>;
```

   Note: You could also unset the default role, thus forcing users to explicitly assume a role every time they log in.

Default Value:

By default, only the user who creates a Snowflake account is assigned the `ACCOUNTADMIN` role as the default role.

References:

1. https://docs.snowflake.com/en/user-guide/security-access-control-configure.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●

1.13 Ensure that the ACCOUNTADMIN or SECURITYADMIN role is not granted to any custom role (Automated)

Profile Applicability:

- Level 1

Description:

The principle of least privilege requires that every identity is only given privileges that are necessary to complete its tasks.

The `ACCOUNTADMIN` system role is the most powerful role in a Snowflake account and is intended for performing initial setup and managing account-level objects. `SECURITYADMIN` role can trivially escalate their privileges to that of `ACCOUNTADMIN`. Neither of these roles should be used for performing daily non-administrative tasks in a Snowflake account.

Rationale:

Granting ACCOUNTADMIN role to any custom role effectively elevates privileges of that role to the `ACCOUNTADMIN` role privileges. Roles that include the `ACCOUNTADMIN` role can then be mistakenly used in access grants that do not require `ACCOUNTADMIN` privileges thus violating the principle of least privilege and increasing the attack surface. The same logic applies to the `SECURITYADMIN` role.

Impact:

Users who lose the `ACCOUNTADMIN` or `SECURITYADMIN` privileges granted to them indirectly through a custom role may not be able to perform their job duties until they regain privileges they legitimately require through a more scoped down role.

Audit:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. List all custom roles granted `ACCOUNTADMIN` or `SECURITYADMIN`:

```
SELECT GRANTEE_NAME AS CUSTOM_ROLE,
  PRIVILEGE AS GRANTED_PRIVILEGE,
  NAME AS GRANTED_ROLE
FROM SNOWFLAKE.ACCOUNT_USAGE.GRANTS_TO_ROLES
WHERE GRANTED_ON = 'ROLE'
  AND NAME IN ('ACCOUNTADMIN','SECURITYADMIN')
  AND DELETED_ON IS NULL;
```

2. Ensure that the query above returns only one row where the `CUSTOM_ROLE` is `ACCOUNTADMIN`, `GRANTED_PRIVILEGE` is `SECURITYADMIN` and `GRANTED_ROLE` is `USAGE`.

Required privileges:
The query requires the `SECURITY_VIEWER` role on the `Snowflake` database.

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI, find all custom roles that are granted `ACCOUNTADMIN` role and revoke that grant.

```
REVOKE SECURITYADMIN ON ACCOUNT FROM ROLE <custom_role>;
REVOKE ACCOUNTADMIN ON ACCOUNT FROM ROLE <custom_role>;
```

Default Value:

By default no custom roles are granted the `ACCOUNTADMIN` role.

References:

1. https://docs.snowflake.com/en/user-guide/security-access-control-configure.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●

1.14 Ensure that Snowflake tasks are not owned by the ACCOUNTADMIN or SECURITYADMIN roles (Automated)

Profile Applicability:

- Level 1

Description:

The `ACCOUNTADMIN` system role is the most powerful role in a Snowflake account and is intended for performing initial setup and managing account-level objects. `SECURITYADMIN` role can trivially escalate their privileges to that of `ACCOUNTADMIN`. Neither of these roles should be used for running Snowflake tasks. A task should be running using a custom role containing only those privileges that are necessary for successful execution of the task.

Snowflake executes tasks with the privileges of the task owner. The role that has `OWNERSHIP` privilege on the task owns the task.

To avoid granting a task inappropriate privileges, the `OWNERSHIP` privilege on the task run as owner should be assigned to a custom role containing only those privileges that are necessary for successful execution of the task.

Rationale:

The principle of least privilege requires that every identity, including service identities, is only given privileges that are necessary to complete its job.

If a threat actor finds a way to influence or hijack the task execution flow, they may be able to exploit privileges given to the task. In the case of an `ACCOUNTADMIN` or `SECURITYADMIN` roles, that may lead to a full account takeover. Additionally, a mistake in the task implementation coupled with excessive privileges may lead to a reliability incident, e.g. accidentally dropping database objects.

Impact:

Existing stored procedures that are owned by the `ACCOUNTADMIN` or `SECURITYADMIN` roles and run with their privileges will need to be updated to use a task specific custom role. If that role does not have all the privileges required by the task, the task execution may fail.

Audit:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. List all tasks owned by the `ACCOUNTADMIN` or `SECURITYADMIN` roles:

```
SELECT NAME AS STORED_PROCEDURE_NAME,
 GRANTED_TO,
 GRANTEE_NAME AS ROLE_NAME,
 PRIVILEGE
FROM SNOWFLAKE.ACCOUNT_USAGE.GRANTS_TO_ROLES
WHERE GRANTED_ON = 'TASK'
  AND DELETED_ON IS NULL
  AND GRANTED_TO = 'ROLE'
  AND PRIVILEGE = 'OWNERSHIP'
  AND GRANTEE_NAME IN ('ACCOUNTADMIN' , 'SECURITYADMIN');
```

2. Ensure that the query above does not return any results.

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. For each task `<task_name>` that runs with `ACCOUNTADMIN` or `SECURITYADMIN` privileges, create a new role `<task_specific_role>` and assign it to the tasks:

```
CREATE ROLE <task_specific_role>;
GRANT OWNERSHIP ON TASK <task_name> TO ROLE <task_specific_role>;
```

2. After creating a new role and granting ownership of each task to it, for each task `<task_name>` that is owned by `ACCOUNTADMIN` or `SECURITYADMIN` roles, ensure all privileges on the tasks are revoked from the roles:

```
REVOKE ALL PRIVILEGES ON TASK <task_name> FROM ROLE ACCOUNTADMIN;
REVOKE ALL PRIVILEGES ON TASK <task_name> FROM ROLE SECURITYADMIN;
```

Default Value:

By default new tasks are granted permissions of the role that was used to create them.

References:

1. https://docs.snowflake.com/en/user-guide/security-access-control-considerations.html
2. https://docs.snowflake.com/en/user-guide/security-access-control-configure.html
3. https://docs.snowflake.com/en/user-guide/tasks-intro.html#task-security

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●

1.15 Ensure that Snowflake tasks do not run with the ACCOUNTADMIN or SECURITYADMIN role privileges (Automated)

Profile Applicability:

- Level 1

Description:

The `ACCOUNTADMIN` system role is the most powerful role in a Snowflake account and is intended for performing initial setup and managing account-level objects. `SECURITYADMIN` role can trivially escalate their privileges to that of `ACCOUNTADMIN`. Neither of these roles should be used for running Snowflake tasks. A task should be running using a custom role containing only those privileges that are necessary for successful execution of the task.

Rationale:

The principle of least privilege requires that every identity, including service identities, is only given privileges that are necessary to complete its job.

If a threat actor finds a way to influence or hijack the task execution flow, they may be able to exploit privileges given to the task. In the case of an `ACCOUNTADMIN` or `SECURITYADMIN` roles, that may lead to a full account takeover. Additionally, a mistake in the task implementation coupled with excessive privileges may lead to a reliability incident, e.g. accidentally dropping database objects.

Impact:

Existing stored procedures that are owned by the `ACCOUNTADMIN` or `SECURITYADMIN` roles and run with their privileges will need to be updated to use a task specific custom role. If that role does not have all the privileges required by the task, the task execution may fail.

Audit:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. List all tasks that have privileges granted to `ACCOUNTADMIN` or `SECURITYADMIN` roles:

```
SELECT NAME AS STORED_PROCEDURE_NAME,
  GRANTED_TO,
  GRANTEE_NAME AS ROLE_NAME,
  PRIVILEGE
FROM SNOWFLAKE.ACCOUNT_USAGE.GRANTS_TO_ROLES
WHERE GRANTED_ON = 'TASK'
  AND DELETED_ON IS NULL
  AND GRANTED_TO = 'ROLE'
  AND GRANTEE_NAME IN ('ACCOUNTADMIN' , 'SECURITYADMIN');
```

2. Ensure that the query above does not return any results.

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. For each task `<task_name>` that runs with `ACCOUNTADMIN` or `SECURITYADMIN` privileges, create a new role `<task_specific_role>` and assign it to the tasks:

```
CREATE ROLE <task_specific_role>;
GRANT OWNERSHIP ON TASK <task_name> TO ROLE <task_specific_role>;
```

2. After creating a new role and granting privileges to each task, ensure all privileges on the tasks are revoked from the `ACCOUNTADMIN` and `SECURITYADMIN` roles:

```
REVOKE ALL PRIVILEGES ON TASK <task_name> FROM ROLE ACCOUNTADMIN;
REVOKE ALL PRIVILEGES ON TASK <task_name> FROM ROLE SECURITYADMIN;
```

Default Value:

By default new tasks are granted permissions of the role that was used to create them.

References:

1. https://docs.snowflake.com/en/user-guide/security-access-control-considerations.html
2. https://docs.snowflake.com/en/user-guide/security-access-control-configure.html
3. https://docs.snowflake.com/en/user-guide/tasks-intro.html#task-security

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●

1.16 Ensure that Snowflake stored procedures are not owned by the ACCOUNTADMIN or SECURITYADMIN roles (Automated)

Profile Applicability:

- Level 1

Description:

The `ACCOUNTADMIN` system role is the most powerful role in a Snowflake account and is intended for performing initial setup and managing account-level objects. `SECURITYADMIN` role can trivially escalate their privileges to that of `ACCOUNTADMIN`. Neither of these roles should be used for running Snowflake stored procedures. A stored procedure should be running using a custom role containing only those privileges that are necessary for successful execution of the stored procedure.

Snowflake executes stored procedures with the privileges of the stored procedure owner or the caller. Role that has `OWNERSHIP` privilege on the stored procedure owns it.

To avoid granting a stored procedure inappropriate privileges, the `OWNERSHIP` privilege on the stored procedure run as owner should be assigned to a custom role containing only those privileges that are necessary for successful execution of the stored procedure.

Rationale:

The principle of least privilege requires that every identity, including service identities, is only given privileges that are necessary to complete its job.

If a threat actor finds a way to influence or hijack the stored procedure execution flow, they may be able to exploit privileges given to the stored procedure. In the case of an `ACCOUNTADMIN` or `SECURITYADMIN` roles, that may lead to a full account takeover. Additionally, a mistake in the stored procedure implementation coupled with excessive privileges may lead to a reliability incident, e.g. accidentally dropping database objects.

Impact:

Existing stored procedures that are owned by the `ACCOUNTADMIN` or `SECURITYADMIN` roles and run with their privileges will need to be updated to use a stored procedure specific custom role. If that role does not have all the privileges required by the stored procedure, the stored procedure execution may fail.

Audit:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. Find all stored procedures that are owned by the `ACCOUNTADMIN` or `SECURITYADMIN` roles.

```
SELECT *
FROM SNOWFLAKE.ACCOUNT_USAGE.PROCEDURES
WHERE DELETED IS NULL
  AND PROCEDURE_OWNER IN ('ACCOUNTADMIN','SECURITYADMIN');
```

2. Ensure that the query above does not return any results.

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. For each stored procedure `<procedure_name>` that runs with `ACCOUNTADMIN` or `SECURITYADMIN` privileges, create a new role `<procedure_specific_role>` and assign it to the stored procedure:

```
CREATE ROLE <procedure_specific_role>;
GRANT OWNERSHIP ON PROCEDURE <procedure_name> TO ROLE
<procedure_specific_role>;
```

2. After creating a new role and granting ownership of each stored procedure to it, for each stored procedure that is owned by `ACCOUNTADMIN` or `SECURITYADMIN` roles, ensure all privileges on the stored procedure are revoked from the roles:

```
REVOKE ALL PRIVILEGES ON PROCEDURE <procedure_name> FROM ROLE
ACCOUNTADMIN;
REVOKE ALL PRIVILEGES ON PROCEDURE <procedure_name> FROM ROLE
SECURITYADMIN;
```

Default Value:

By default stored procedures that run as owner are granted permissions of the role that was used to create them.

References:

1. https://docs.snowflake.com/en/user-guide/security-access-control-considerations.html
2. https://docs.snowflake.com/en/user-guide/security-access-control-configure.html
3. https://docs.snowflake.com/en/sql-reference/stored-procedures-rights

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●

1.17 Ensure Snowflake stored procedures do not run with ACCOUNTADMIN or SECURITYADMIN role privileges (Automated)

Profile Applicability:

- Level 1

Description:

The `ACCOUNTADMIN` system role is the most powerful role in a Snowflake account; it is intended for performing initial setup and managing account-level objects. Users and stored procedures with the `SECURITYADMIN` role can escalate their privileges to `ACCOUNTADMIN`.

Snowflake stored procedures should not run with the `ACCOUNTADMIN` or `SECURITYADMIN` roles. Instead, stored procedures should be run using a custom role containing only those privileges that are necessary for successful execution of the stored procedure.

Rationale:

The principle of least privilege requires that every identity, including service identities, is only given privileges that are necessary to complete its job.

If a threat actor finds a way to influence or hijack the stored procedure execution flow, they may be able to exploit privileges given to the stored procedure. In the case of an `ACCOUNTADMIN` or `SECURITYADMIN` roles, that may lead to a full account takeover. Additionally, a mistake in the stored procedure implementation coupled with excessive privileges may lead to a reliability incident, e.g. accidentally dropping database objects.

Impact:

Existing stored procedures that are owned by the ACCOUNTADMIN or SECURITYADMIN roles and run with their privileges will need to be updated to use a stored procedure specific custom role. If that role does not have all the privileges required by the stored procedure, the stored procedure execution may fail.

Audit:

Programmatically:
In a Snowsight worksheet or using the SnowSQL CLI:

1. Find the list of stored procedures that run with `ACCOUNTADMIN` or `SECURITYADMIN` role privileges:

```
SELECT NAME AS STORED_PROCEDURE_NAME,
  GRANTED_TO,
  GRANTEE_NAME AS ROLE_NAME
FROM SNOWFLAKE.ACCOUNT_USAGE.GRANTS_TO_ROLES
WHERE GRANTED_ON = 'PROCEDURE'
  AND DELETED_ON IS NULL
  AND GRANTED_TO = 'ROLE'
  AND GRANTEE_NAME IN ('ACCOUNTADMIN' , 'SECURITYADMIN');
```

2. Ensure that the query above does not return any results.

Remediation:

Programmatically:
In a Snowsight worksheet or using the SnowSQL CLI:

1. For each stored procedure `<procedure_name>` that runs with `ACCOUNTADMIN` or `SECURITYADMIN` privileges, create a new role `<procedure_specific_role>` and assign it to the stored procedure:

```
CREATE ROLE <procedure_specific_role>;
GRANT OWNERSHIP ON PROCEDURE <procedure_name> TO ROLE
<procedure_specific_role>;
```

2. After creating a new role and granting privileges to each stored procedure, ensure all privileges on the stored procedure `<procedure_name>` are revoked from the `ACCOUNTADMIN` and `SECURITYADMIN` roles:

```
REVOKE ALL PRIVILEGES ON PROCEDURE <procedure_name> FROM ROLE
ACCOUNTADMIN;
REVOKE ALL PRIVILEGES ON PROCEDURE <procedure_name> FROM ROLE
SECURITYADMIN;
```

Default Value:

By default stored procedures that run as owner are granted permissions of the role that was used to create them.

References:

1. https://docs.snowflake.com/en/user-guide/security-access-control-considerations.html
2. https://docs.snowflake.com/en/user-guide/security-access-control-configure.html
3. https://docs.snowflake.com/en/sql-reference/stored-procedures-rights

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●

2 Monitoring and Alerting

This section contains recommendations for configuring Snowflake to assist with monitoring and responding to account activities.

All remediation sections use a notification integration. You can define with the following SQL command:

1. To create a notification integration:

```
CREATE NOTIFICATION INTEGRATION cis_monitoring_notifications
  TYPE=EMAIL
  ENABLED=TRUE
  ALLOWED_RECIPIENTS=('your_email');
```

   Note: You can define a maximum of ten email notification integrations for a given account. Therefore, it's best to use the same notification integration for all the alert and monitoring tasks.

6. To send an email notification to an email address using that notification integration:

```
CALL SYSTEM$SEND_EMAIL(
  'my_email_int',
  'your_email',
        'Email Alert: Task A has finished.',
        'Task A has successfully finished.\nStart Time: 10:10:32\nEnd Time: 12:15:45\nTotal Records Processed: 115678'
  );
```

13. To create a task that runs every hour and send a monitoring email:

```
 CREATE OR REPLACE TASK monitoring_task
        WAREHOUSE = 'warehouse_name'
        SCHEDULE = '60 MINUTE'
        COMMENT = 'Task to monitor and alert on security events'
        AS
        BEGIN
          -- Configure your task here
          CALL SYSTEM$SEND_EMAIL(
          'my_email_int',
          'your_email',
          'Email Alert: Task A has finished.',
          'Task A has successfully finished.\nStart Time: 10:10:32\nEnd Time: 12:15:45\nTotal Records Processed: 115678'
 );
 END;
```

For more information, see the documentation on email notifications on Snowflake.

For more information on creating a task, follow the instructions on the Create task documentation.

2.1 Ensure monitoring and alerting exist for ACCOUNTADMIN and SECURITYADMIN role grants (Manual)

Profile Applicability:

- Level 1

Description:

By default, `ACCOUNTADMIN` is the most powerful role in a Snowflake account and users with `SECURITYADMIN` role grant can trivially escalate their privileges to that of `ACCOUNTADMIN`.

Following the principle of least privilege that prescribes limiting user's privileges to those that are strictly required to do their jobs, the `ACCOUNTADMIN` and `SECURITYADMIN` roles should be assigned to a limited number of designated users. Any new `ACCOUNTADMIN` and `SECURITYADMIN` role grants should be scrutinized.

Rationale:

Every new `ACCOUNTADMIN` and `SECURITYADMIN` role assignment increases the attack surface of a Snowflake environment. It may also indicate unauthorized privilege escalation performed by a threat actor.

If monitoring for `ACCOUNTADMIN` role assignments is not configured, inappropriate or unauthorized `ACCOUNTADMIN` role access grants may be missed. The latter can lead to eventual security posture degradation or late detection of an ongoing security incident. The same logic applies to the `SECURITYADMIN` role.

Impact:

If the principle of least privilege is not strictly applied and `ACCOUNTADMIN` and `SECURITYADMIN` role assignments happen frequently, monitoring and alerting on this event may generate undue load on the detection and response team.

Audit:

Confirm that your security monitoring and alerting solution is configured to generate alerts on new `ACCOUNTADMIN` and `SECURITYADMIN` role assignments.

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. Configure your monitoring task to alert on `ACCOUNTADMIN` and `SECURITYADMIN` role grants. You can find those grants with the following query:

```
SELECT *
FROM SNOWFLAKE.ACCOUNT_USAGE.GRANTS_TO_ROLES
WHERE NAME IN ('ACCOUNTADMIN', 'SECURITYADMIN');
```

Default Value:

There is no `ACCOUNTADMIN` or `SECURITYADMIN` role assignment event monitoring and alerting by default.

References:

1. https://docs.snowflake.com/en/user-guide/ui-snowsight-activity.html
2. https://docs.snowflake.com/en/user-guide/security-access-control-considerations.html
3. https://docs.snowflake.com/en/sql-reference/sql/create-notification-integration
4. https://docs.snowflake.com/en/sql-reference/sql/create-task;
5. https://docs.snowflake.com/en/user-guide/email-stored-procedures

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.11 Conduct Audit Log Reviews Conduct reviews of audit logs to detect anomalies or abnormal events that could indicate a potential threat. Conduct reviews on a weekly, or more frequent, basis.		●	●
v7	6.7 Regularly Review Logs On a regular basis, review logs to identify anomalies or abnormal events.		●	●

2.2 Ensure monitoring and alerting exist for MANAGE GRANTS privilege grants (Manual)

Profile Applicability:

- Level 1

Description:

The `MANAGE GRANTS` privilege is one of the most powerful privileges in the Snowflake environment. This privilege gives the ability to grant or revoke privileges on any object as if the invoking role were the owner of the object.

A custom role with the `MANAGE GRANTS` privilege on account level will not be able to grant privileges on the account level as that privilege is implicitly reserved for the `ACCOUNTADMIN` and `SECURITYADMIN` roles. However, such custom roles will be able to grant any privileges on any objects below the account level.

Following the principle of least privilege and given how powerful the `MANAGE GRANTS` privilege is, any new `MANAGE GRANTS` privilege grants should be scrutinized.

Rationale:

Every new role granted the `MANAGE GRANTS` privilege increases the attack surface of a Snowflake environment. It may also indicate unauthorized privilege escalation performed by a threat actor.

If monitoring for `MANAGE GRANTS` privilege grants is not configured, inappropriate or unauthorized `MANAGE GRANTS` privilege grants may be missed. The latter can lead to eventual security posture degradation or late detection of an ongoing security incident.

Impact:

If `MANAGE GRANTS` privilege grants happen frequently, monitoring and alerting on this event may generate undue load on the detection and response team.

Audit:

Confirm that your security monitoring and alerting solution is configured to generate alerts on new `MANAGE GRANTS` privilege grants.

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. Configure your monitoring task to alert on manage grants privilege grants.

```
select end_time, query_type
  query_text,
  user_name,
  role_name
from snowflake.account_usage.query_history
where execution_status = 'SUCCESS'
  and query_type = 'GRANT'
  and regexp_instr(query_text, 'manage\\s*grants', 1, 1, 0, 'i') > 0
  order by end_time desc;
```

Default Value:

There is no MANAGE GRANTS privilege grant event monitoring and alerting set up by default.

References:

1. https://docs.snowflake.com/en/user-guide/ui-snowsight-activity.html
2. https://docs.snowflake.com/en/user-guide/security-access-control-privileges.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.11 Conduct Audit Log Reviews Conduct reviews of audit logs to detect anomalies or abnormal events that could indicate a potential threat. Conduct reviews on a weekly, or more frequent, basis.		●	●
v7	6.7 Regularly Review Logs On a regular basis, review logs to identify anomalies or abnormal events.		●	●

2.3 Ensure monitoring and alerting exist for password sign-ins of SSO users (Manual)

Profile Applicability:

- Level 1

Description:

The security benefit of SSO is to relieve users from having to set up and manage distinct sets of credentials for distinct applications and services. It also allows security administrators to focus on hardening and defending only one identity storage and limited number of user credentials.

Rationale:

Allowing users to sign in with Snowflake passwords in the presence of a configured third-party identity provider SSO may undermine mandatory security controls configured on the SSO and degrade security posture of the account. For example, the SSO sign-in flow may be configured to require multi-factor authentication (MFA), where Snowflake password sign-in flow may not.

Every Snowflake password-based sign-in may indicate an unapproved authentication flow taking place.

Impact:

If password sign-in events happen frequently, monitoring and alerting on this event may generate undue load on the detection and response team.

Audit:

Confirm that your security monitoring and alerting solution is configured to generate alerts on new password sign-in events.

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. Configure your security monitoring solution to alert on password sign-ins of SSO users. The following query can be run periodically.

```
select event_timestamp,
  user_name,
  client_ip,
  reported_client_type,
  reported_client_version,
  first_authentication_factor,
  second_authentication_factor
from snowflake.account_usage.login_history
where first_authentication_factor = 'PASSWORD'
order by event_timestamp desc;
```

Default Value:

There is no password sign-in event monitoring and alerting by default.

References:

1. https://docs.snowflake.com/en/user-guide/ui-snowsight-activity.html
2. https://docs.snowflake.com/en/user-guide/admin-security-fed-auth.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.11 Conduct Audit Log Reviews Conduct reviews of audit logs to detect anomalies or abnormal events that could indicate a potential threat. Conduct reviews on a weekly, or more frequent, basis.		●	●
v7	6.7 Regularly Review Logs On a regular basis, review logs to identify anomalies or abnormal events.		●	●

2.4 Ensure monitoring and alerting exist for password sign-in without MFA (Manual)

Profile Applicability:

- Level 1

Description:

Multi-factor authentication (MFA) is a security control used to add an additional layer of login security. It works by requiring the user to present two or more proofs (factors) of user identity. An MFA example would be requiring a password and a verification code delivered to the user's phone during user sign-in.

The MFA feature for Snowflake users is powered by the Duo Security service.

Rationale:

MFA mitigates security threats of users creating weak passwords and user passwords being stolen or accidentally leaked.

Impact:

If password sign-in events without MFA happen frequently, monitoring and alerting on this event may generate undue load on the detection and response team.

Audit:

Confirm that your security monitoring and alerting solution is configured to generate alerts on new password sign-in without MFA events.

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. Configure your security monitoring solution to alert on password sign-ins without MFA. The following query can be run periodically.

```
select event_timestamp,
  user_name,
  client_ip,
  reported_client_type,
  reported_client_version,
  first_authentication_factor,
  second_authentication_factor
from snowflake.account_usage.login_history
where first_authentication_factor = 'PASSWORD'
  and second_authentication_factor is null
order by event_timestamp desc;
```

Default Value:

There is no password sign-in without MFA event monitoring and alerting by default.

References:

1. https://docs.snowflake.com/en/user-guide/security-access-control-configure.html
2. https://docs.snowflake.com/en/user-guide/security-mfa.html
3. https://docs.snowflake.com/en/user-guide/ui-snowsight-activity.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.11 Conduct Audit Log Reviews Conduct reviews of audit logs to detect anomalies or abnormal events that could indicate a potential threat. Conduct reviews on a weekly, or more frequent, basis.		●	●
v7	6.7 Regularly Review Logs On a regular basis, review logs to identify anomalies or abnormal events.		●	●

2.5 Ensure monitoring and alerting exist for creation, update and deletion of security integrations (Manual)

Profile Applicability:

- Level 1

Description:

Security integration object is used to configure SSO and SCIM integrations.

Rationale:

Creation of an unauthorized security integration, in case of SCIM, can lead to creation of rogue Snowflake users. Incase of SSO, it can lead to hijacking of existing Snowflake users through rogue authentication flow.

Update or deletion of an existing security integration can lead to weakening security posture of that integration or denial of service, e.g. when users cannot sign into Snowflake accounts due to broken SSO authentication flow.

Impact:

If security integration creation, update and deletion events happen frequently, monitoring and alerting on this event may generate undue load on the detection and response team.

Audit:

Confirm that your security monitoring and alerting solution is configured to generate alerts on new security integration creation, update and deletion events.

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. Configure your security monitoring solution to alert on creation, update and deletion of security integrations.

```
select end_time,
  query_type,
  query_text,
  user_name,
  role_name
from snowflake.account_usage.query_history
where execution_status = 'SUCCESS'
  and query_type in ('CREATE', 'ALTER', 'DROP')
  and query_text ilike '%security integration%'
order by end_time desc;
```

Default Value:

There is no security integration creation, update and deletion event monitoring and alerting by default.

References:

1. https://docs.snowflake.com/en/sql-reference/sql/create-security-integration.html
2. https://docs.snowflake.com/en/user-guide/ui-snowsight-activity.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.11 Conduct Audit Log Reviews Conduct reviews of audit logs to detect anomalies or abnormal events that could indicate a potential threat. Conduct reviews on a weekly, or more frequent, basis.		●	●
v7	6.7 Regularly Review Logs On a regular basis, review logs to identify anomalies or abnormal events.		●	●

2.6 Ensure monitoring and alerting exist for changes to network policies and associated objects (Manual)

Profile Applicability:

- Level 1

Description:

Network policies allow restricting access to a Snowflake account based on source IP addresses. A network policy can be configured either on the account level, for all users of the account, or on the user level, for a specific user. In the presence of both account-level and user-level policies the latter takes precedence.

A network policy can also be configured on the SCIM and Snowflake OAuth security integrations to restrict the list of source IP addresses allowed when exchanging an authorization code for an access or refresh token and when using a refresh token to obtain a new access token. If network policy is not set on the security integration of the aforementioned types, the account-level network policy, if any, is used.

Rationale:

Creation and application of unauthorized network policies can weaken access control through expansion of the allowed source IP addresses, or lead to a denial of service through blocklisting legitimate source IP addresses. Unauthorized changes and deletions of existing network policies can lead to the same undesirable results.

Impact:

If network policy creation, update, deletion and object association events happen frequently, monitoring and alerting on this event may generate undue load on the detection and response team.

Audit:

Confirm that your security monitoring and alerting solution is configured to generate alerts on new security integration creation, update and deletion events.

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. Configure your security monitoring solution to alert on changes to network policies.

```
select end_time,
  query_type,
  query_text,
  user_name,
  role_name
from snowflake.account_usage.query_history
where execution_status = 'SUCCESS'
  and (
  query_type in ('CREATE_NETWORK_POLICY', 'ALTER_NETWORK_POLICY',
  'DROP_NETWORK_POLICY')
  or (query_text ilike '%set%network_policy%'
  or query_text ilike '%unset%network_policy%')
)
order by end_time desc;
```

Default Value:

There is no network policy creation, update, deletion or object association event monitoring and alerting by default.

References:

1. https://docs.snowflake.com/en/user-guide/network-policies.html
2. https://docs.snowflake.com/en/sql-reference/sql/create-security-integration-oauth-snowflake.html
3. https://docs.snowflake.com/en/sql-reference/sql/create-security-integration-scim.html
4. https://docs.snowflake.com/en/user-guide/scim-custom.html#limitations
5. https://docs.snowflake.com/en/user-guide/ui-snowsight-activity.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.11 Conduct Audit Log Reviews Conduct reviews of audit logs to detect anomalies or abnormal events that could indicate a potential threat. Conduct reviews on a weekly, or more frequent, basis.		●	●
v7	6.7 Regularly Review Logs On a regular basis, review logs to identify anomalies or abnormal events.		●	●

2.7 Ensure monitoring and alerting exist for SCIM token creation (Manual)

Profile Applicability:

- Level 1

Description:

The System for Cross-domain Identity Management (SCIM) is an open specification designed to help facilitate the automated management of user identities and groups (i.e. roles) in cloud applications using RESTful APIs.

Snowflake supports SCIM 2.0 integration with Okta, Microsoft Azure AD and custom identity providers. Users and groups from the identity provider can be provisioned into Snowflake, which functions as the service provider.

SCIM access token is a bearer token used by SCIM clients to authenticate to Snowflake SCIM server.

Rationale:

SCIM access tokens generated without proper authorization may be used for configuring rogue SCIM integrations. Such SCIM integrations can then be used for provisioning rogue users that through existing roles are granted unauthorized access to Snowflake data and other objects.

Impact:

If SCIM access token creation events happen frequently, monitoring and alerting on this event may generate undue load on the detection and response team. That said, a SCIM access token is valid for 6 months and there is usually only one SCIM integration per account. Frequent SCIM access token creation would likely be an unusual event.

Audit:

Confirm that your security monitoring and alerting solution is configured to generate alerts on new security SCIM access token events.

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. Configure your security monitoring solution to alert on SCIM token creation. The following query can be run periodically.

```
select end_time,
  query_type,
  query_text,
  user_name,
  role_name
from snowflake.account_usage.query_history
where execution_status = 'SUCCESS'
         and query_type = 'SELECT'
         and regexp_instr(query_text,
'system\\$generate_scim_access_token\\s*\\(', 1, 1, 0, 'i') > 0
order by end_time desc;
```

Default Value:

There is no SCIM access token creation event monitoring and alerting by default.

References:

1. https://docs.snowflake.com/en/user-guide/scim-intro.html#scim-overview
2. https://docs.snowflake.com/en/user-guide/scim-custom.html#create-a-custom-scim-security-integration-and-api-token
3. https://docs.snowflake.com/en/user-guide/ui-snowsight-activity.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.11 Conduct Audit Log Reviews Conduct reviews of audit logs to detect anomalies or abnormal events that could indicate a potential threat. Conduct reviews on a weekly, or more frequent, basis.		●	●
v7	6.7 Regularly Review Logs On a regular basis, review logs to identify anomalies or abnormal events.		●	●

2.8 Ensure monitoring and alerting exists for new share exposures (Manual)

Profile Applicability:

- Level 1

Description:

Snowflake tables, views and UDFs can be shared across Snowflake accounts using share objects created by data providers and imported by data consumers.

To expose a share to another account, the share provider account needs to add or set consumer accounts on a share using the `ALTER SHARE` command. The consumer account can then import the share using the `CREATE DATABASE FROM SHARE` command.

Rationale:

A share exposed to another Snowflake account can be used for data exfiltration.

Impact:

If exposing shares to another account event happens frequently, monitoring and alerting on this event may generate undue load on the detection and response team.

Audit:

Confirm that your security monitoring and alerting solution is configured to generate alerts on new share exposures.

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. Configure your security monitoring solution to alert on new share exposures. The following query can be run periodically.

```
select end_time,
  query_type,
  query_text,
  user_name,
  role_name
from snowflake.account_usage.query_history
where execution_status = 'SUCCESS'
         and query_type = 'ALTER'
         and regexp_instr(query_text,
'^alter\\s*share.*(add|set)\\s*accounts\\s*=', 1, 1, 0, 'is') > 0
order by end_time desc;
```

Default Value:

There is no share exposure event monitoring and alerting by default.

References:

1. https://docs.snowflake.com/en/user-guide/data-sharing-intro
2. https://docs.snowflake.com/en/sql-reference/sql/alter-share

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.11 Conduct Audit Log Reviews Conduct reviews of audit logs to detect anomalies or abnormal events that could indicate a potential threat. Conduct reviews on a weekly, or more frequent, basis.		●	●
v7	6.7 Regularly Review Logs On a regular basis, review logs to identify anomalies or abnormal events.		●	●

2.9 Ensure monitoring and alerting exists for sessions from unsupported Snowflake Connector for Python and JDBC and ODBC drivers (Manual)

Profile Applicability:

- Level 2

Description:

Snowflake provides client software (drivers, connectors, etc.) for connecting to Snowflake and using certain Snowflake features (e.g. Apache Kafka for loading data, Apache Hive metadata for external tables). The clients must be installed on each local workstation or system from which you wish to connect. The Snowflake Connector for Python, JDBC and ODBC drivers are some of the most used Snowflake clients.

Old versions of drivers and connectors may contain security vulnerabilities that have been fixed in the latest version. To ensure that only up-to-date software is used, you should actively monitor session logins coming from unsupported clients and upgrade those to the latest available versions.

Rationale:

Using out-of-date Snowflake clients can expose your account to security risks. You should monitor for connections from unsupported Snowflake Connector for Python and JDBC and ODBC drivers and upgrade to the latest versions available.

Impact:

None.

Audit:

Confirm that your security monitoring and alerting solution is configured to generate alerts on sessions coming from unsupported Snowflake connectors and drivers.

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. Check the Recommended Client Versions documentation and note the minimum versions of the Snowflake Connector for Python, JDBC driver and ODBC driver.
2. Create a UDF to help you compare version numbers:

```
CREATE OR REPLACE FUNCTION compare_versions(v1 VARCHAR, v2 VARCHAR)
-- result compares v1 and v2
-- result == lower means that v1 is lower than v2
  RETURNS VARCHAR
  AS
  $$
    case
        when CAST(SPLIT(v1, '.')[0] AS NUMBER) < CAST(SPLIT(v2, '.')[0]
AS NUMBER) then 'lower'
        when CAST(SPLIT(v1, '.')[0] AS NUMBER) > CAST(SPLIT(v2, '.')[0]
AS NUMBER) then 'higher'
        when CAST(SPLIT(v1, '.')[1] AS NUMBER) < CAST(SPLIT(v2, '.')[1]
AS NUMBER) then 'lower'
        when CAST(SPLIT(v1, '.')[1] AS NUMBER) > CAST(SPLIT(v2, '.')[1]
AS NUMBER) then 'higher'
        when CAST(SPLIT(v1, '.')[2] AS NUMBER) < CAST(SPLIT(v2, '.')[2]
AS NUMBER) then 'lower'
        when CAST(SPLIT(v1, '.')[2] AS NUMBER) > CAST(SPLIT(v2, '.')[2]
AS NUMBER) then 'higher'
        else 'equal'
    end
  $$
  ;
```

3. Configure your security monitoring solution to alert on sessions from unsupported versions. Replace the version numbers below with the latest versions from the previous step. The following query can be run periodically.

```
SELECT CREATED_ON, USER_NAME,
  SPLIT(CLIENT_APPLICATION_ID, ' ')[0]::varchar AS "CLIENT_APP",
  CLIENT_APPLICATION_VERSION,
  CLIENT_ENVIRONMENT
FROM SNOWFLAKE.ACCOUNT_USAGE.SESSIONS
WHERE ("CLIENT_APP" = 'JDBC' AND
COMPARE_VERSIONS(CLIENT_APPLICATION_VERSION, '3.13.6') = 'lower')
  OR ("CLIENT_APP" = 'ODBC' AND
COMPARE_VERSIONS(CLIENT_APPLICATION_VERSION, '2.23.3') = 'lower')
  OR ("CLIENT_APP" = 'PythonConnector' AND
COMPARE_VERSIONS(CLIENT_APPLICATION_VERSION, '2.5.0') = 'lower')
ORDER BY CLIENT_APPLICATION_ID;
```

4. When detecting the use of unsupported clients, upgrade to the latest, recommended version.

Default Value:

By default, there is no monitoring and alerting on unsupported clients.

References:

1. https://docs.snowflake.com/en/release-notes/requirements
2. https://community.snowflake.com/s/article/how-to-report-on-the-clients-connecting-to-a-snowflake-account

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.11 Conduct Audit Log Reviews Conduct reviews of audit logs to detect anomalies or abnormal events that could indicate a potential threat. Conduct reviews on a weekly, or more frequent, basis.		●	●
v7	6.7 Regularly Review Logs On a regular basis, review logs to identify anomalies or abnormal events.		●	●

3 Networking

This section contains recommendations for configuring security-related aspects of Snowflake networking.

3.1 Ensure that an account-level network policy has been configured to only allow access from trusted IP addresses (Manual)

Profile Applicability:

- Level 2

Description:

Network policies allow restricting access to a Snowflake account based on source IP addresses. A network policy can be configured either on the account level, for all users of the account, or on the user level, for a specific user. In the presence of both account-level and user-level policies, the user-level policies take precedence.

A network policy can also be configured on the SCIM and Snowflake OAuth security integrations to restrict the list of source IP addresses allowed when exchanging an authorization code for an access or refresh token and when using a refresh token to obtain a new access token. If network policy is not set on the security integration of the aforementioned types, the account-level network policy is set, if used.

Rationale:

Network policies help mitigate the threat of leaked user credentials. If an account network policy is not configured limiting source IP addresses, leaked Snowflake credentials can be used from anywhere in the world.

Network policies are especially useful when there is a heightened risk of leaking credentials. For example, if instead of using SSO, users authenticate to Snowflake using Snowflake passwords.

Network policy set on the account level can serve as a coarse-grained baseline for the majority of the Snowflake users and can be further tightened on the specific highly privileged user, service account, and security integration level.

Impact:

If a network policy is misconfigured to disallow IP addresses from which users usually access Snowflake, their productivity may be impacted.

If a network policy is misconfigured to disallow IP addresses from which services and automation usually access Snowflake, reliability of those services and automation may be impacted.

If a network policy is misconfigured to disallow IP addresses used by one of the Snowflake security integrations that support network policies, those integrations will be broken.

If a user with permissions to configure network policies on the account accidentally locks themselves and everybody else with such permission out, they will need to contact Snowflake customer support to restore access to their account.

Audit:

From the UI:

1. Go to https://app.snowflake.com/ and sign into your Snowflake account.
2. On the left side navigation bar, click on *Admin*.
3. Under *Admin*, click on *Security*.
4. Under the *Network Policies* tab, ensure that a network policy is configured properly and set to `Active`.

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. List the network policies active at the account level.

```
SHOW PARAMETERS LIKE 'NETWORK_POLICY' IN ACCOUNT;
```

2. Ensure that the query returns a result.
3. Note the name of the network policy and replace `<policy_name>` in the query below with your network policy name:

```
DESCRIBE NETWORK POLICY <policy_name>;
```

4. Ensure that `ALLOWED_IP_LIST` is set for the network policy and that it contains only trusted IP address ranges.

Remediation:

From the UI:

1. Go to https://app.snowflake.com/ and sign into your Snowflake account.
2. On the left side navigation bar, click on *Admin*.
3. Under *Admin*, click on *Security*.
4. Under the *Network Policies* tab, click the `+ Network Policy` button on the top right side.
5. Enter a `Policy Name` and list of `Allowed IP Addresses`.
6. Click `Create network policy`.
7. Find your policy in the list of network policies and click `Activate policy`. This will set the network policy at the account level.

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. Create a network policy. Replace `<policy_name>` with the name you want to give the policy, and customize the list of allowed and blocked IP addresses:

```
CREATE NETWORK POLICY <policy_name> ALLOWED_IP_LIST=('192.168.1.0/24');
```

2. Set the network policy at the account level:

```
ALTER ACCOUNT SET NETWORK_POLICY = <policy_name>;
```

For more information, see the documentation on creating network policies.
Note:

- When a network policy includes values for both `ALLOWED_IP_LIST` and `BLOCKED_IP_LIST`, Snowflake applies the blocked list first.
- Do not add `0.0.0.0/0` to `BLOCKED_IP_LIST`. Because Snowflake applies the blocked list first, this would block your own access. Additionally, in order to block all IP addresses except a select list, you only need to add IP addresses to `ALLOWED_IP_LIST`. Snowflake automatically blocks all IP addresses not included in the allowed list.
- You can create and set a network policy on a security integration to configure allowed IP addresses from your IdP used to exchange an authorization code for an access or refresh token and when using a refresh token to obtain a new access token.

Default Value:

No network policies are configured by default. Access from any IP address is allowed.

References:

1. https://docs.snowflake.com/en/user-guide/network-policies.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.4 Implement and Manage a Firewall on Servers Implement and manage a firewall on servers, where supported. Example implementations include a virtual firewall, operating system firewall, or a third-party firewall agent.	●	●	●
v7	9.4 Apply Host-based Firewalls or Port Filtering Apply host-based firewalls or port filtering tools on end systems, with a default-deny rule that drops all traffic except those services and ports that are explicitly allowed.	●	●	●

3.2 Ensure that user-level network policies have been configured for service accounts (Manual)

Profile Applicability:

- Level 1

Description:

Network policies allow restricting access to a Snowflake account based on source IP addresses. A network policy can be configured either on the account level, for all users of the account, or on the user level, for a specific user. In the presence of both account-level and user-level policies, the user-level policies take precedence.

A service account is a Snowflake user whose credentials are used by scripts, jobs, applications, pipelines, etc. to talk to Snowflake. Other names include "application user", "service principal", "system account", or "daemon user". Service account is not a Snowflake specific term.

Rationale:

Network policies help mitigate the threat of leaked user credentials. If network policies are not configured limiting source IP addresses, leaked Snowflake credentials can be used from anywhere in the world.

Service accounts often have direct access to raw sensitive data not appropriate for most human users. Service accounts are also generally deployed in production environments with source IP address ranges distinct from the IP address ranges used by the human users. To decrease the risk of inappropriate data access with service account credentials, user-level network policies can be applied to service accounts.

Impact:

If a network policy is misconfigured to disallow IP addresses from which service accounts access Snowflake, it can cause a reliability impact.

If a user with permissions to configure network policies on the account accidentally locks themselves and everybody else with such permission out, they will need to contact Snowflake customer support to restore access to their account.

Audit:

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. Get the list of users in the account.

```
SELECT object_name
FROM snowflake.account_usage.users
WHERE domain = 'USER' AND u.deleted_on IS NULL;
```

2. Identify the users that are used to run tasks, also known as service accounts.
3. For each service account `<service_account_name>`, check if there is a network policy associated with it:

```
SHOW PARAMETERS LIKE 'NETWORK_POLICY' FOR USER <service_account_name>;
```

   Note: The name of the network policy from the `value` field.

4. Describe the policy. Replace `<policy_name>` in the query below with your network policy name from above:

```
DESCRIBE NETWORK POLICY <policy_name>;
```

5. Ensure that `ALLOWED_IP_LIST` is set for the network policy and that it contains only trusted IP address ranges.

Required Privileges:
To run the queries above, the caller needs:

- `OWNERSHIP` privilege on every network policy in an account.
- `SECURITY_VIEWER` role on the `Snowflake` database
- `GOVERNANCE_VIEWER` role on the `Snowflake` database

Remediation:

Programmatically:
In a Snowsight worksheet or through the SnowSQL ALI:

1. Create a network policy. Replace `<policy_name>` with the name you want to give the policy, and customize the list of allowed and blocked IP addresses:

```
CREATE NETWORK POLICY <policy_name> ALLOWED_IP_LIST=('192.168.1.0/24');
```

2. For each service account user `<service_account_name>`, set the desired network policy `<policy_name>`:

```
ALTER USER <service_account_name>
    SET NETWORK_POLICY = <policy_name>;
```

For more information, see the documentation on creating network policies.
Note:

- When a network policy includes values for both `ALLOWED_IP_LIST` and `BLOCKED_IP_LIST`, Snowflake applies the blocked list first.
- Do not add `0.0.0.0/0` to `BLOCKED_IP_LIST`. Because Snowflake applies the blocked list first, this would block your own access. Additionally, in order to block all IP addresses except a select list, you only need to add IP addresses to `ALLOWED_IP_LIST`. Snowflake automatically blocks all IP addresses not included in the allowed list.

Default Value:

No network policies are configured by default for any user. Access from any IP address is allowed.

References:

1. https://docs.snowflake.com/en/user-guide/network-policies.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.4 Implement and Manage a Firewall on Servers Implement and manage a firewall on servers, where supported. Example implementations include a virtual firewall, operating system firewall, or a third-party firewall agent.	●	●	●
v7	9.4 Apply Host-based Firewalls or Port Filtering Apply host-based firewalls or port filtering tools on end systems, with a default-deny rule that drops all traffic except those services and ports that are explicitly allowed.	●	●	●

4 Data Protection

This section contains recommendations for configuring data protection in Snowflake.

4.1 Ensure yearly rekeying is enabled for a Snowflake account (Automated)

Profile Applicability:

- Level 2

Description:

All Snowflake customer data is encrypted by default using the latest security standards and best practices. Snowflake uses strong AES 256-bit encryption with a hierarchical key model rooted in a hardware security module.

All Snowflake-managed keys are automatically rotated when they are more than 30 days old. Furthermore, data can be automatically re-encrypted ("rekeyed") on a yearly basis. Data encryption and key rotation is entirely transparent and requires no configuration or management.

Key rotation transitions an active encryption key to a retired state. Practically this means transitioning of the active encryption key from being used for encrypting new data and decrypting data encrypted with that key to only decrypting data encrypted with that key.

Rekeying transitions a retired encryption key to being destroyed. Practically this means re-encryption of the data encrypted by a retired key with a new key and destroying the disposing of the retired key.

Rationale:

Rekeying constrains the total duration in which a key is used for recipient usage, following NIST recommendations. Furthermore, when rekeying data, Snowflake can increase encryption key sizes and utilize better encryption algorithms that may be standardized since the previous key generation was created.

Rekeying, therefore, ensures that all customer data, new and old, is encrypted with the latest security technology.

Impact:

None.

Audit:

Programmatically:
In a Snowsight worksheet or from the SnowSQL CLI:

1. List the value of the `PERIODIC_DATA_REKEYING` parameter:

   ```
   SHOW PARAMETERS LIKE 'PERIODIC_DATA_REKEYING' IN ACCOUNT;
   ```

2. Ensure that the parameter is set to `true`.

Remediation:

Programmatically:
Set parameter value to `true`:

```
ALTER ACCOUNT
    SET PERIODIC_DATA_REKEYING=true;
```

Default Value:

By default, yearly re-keying is disabled.

References:

1. https://docs.snowflake.com/en/user-guide/security-encryption-manage.html
2. https://docs.snowflake.com/en/sql-reference/parameters.html#periodic-data-rekeying

Additional Information:

Periodic data rekeying setting is only available to Enterprise Edition or higher.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.11 Encrypt Sensitive Data at Rest Encrypt sensitive data at rest on servers, applications, and databases containing sensitive data. Storage-layer encryption, also known as server-side encryption, meets the minimum requirement of this Safeguard. Additional encryption methods may include application-layer encryption, also known as client-side encryption, where access to the data storage device(s) does not permit access to the plain-text data.		●	●
v7	14.8 Encrypt Sensitive Information at Rest Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.			●

4.2 Ensure AES encryption key size used to encrypt files stored in internal stages is set to 256 bits (Automated)

Profile Applicability:

- Level 1

Description:

All ingested data stored in Snowflake tables is encrypted using 256-bit long AES encryption keys. However, data uploaded to internal stages is by default encrypted with 128-bit long AES encryption keys.

Rationale:

The field of cryptanalysis is continuously advancing and new vulnerabilities and attacks are discovered that obsolete cryptographic primitives that once were considered secure.

The 128-bit long AES encryption keys are still considered secure today and there are no strong reasons to believe this will change soon. Usage of the 256-bit long AES encryption keys today is generally recommended out of an abundance of caution.

Impact:

None.

Audit:

Programmatically:
In a Snowsight worksheet or using the SnowSQL CLI:

1. Check length of the AES encryption keys used to encrypt data uploaded to internal stages:

```
SHOW PARAMETERS LIKE 'CLIENT_ENCRYPTION_KEY_SIZE' IN ACCOUNT;
```

2. Ensure that `value` is set to `256`.

Remediation:

Programmatically:
To set the length of the AES encryption keys used to encrypt data uploaded to internal stages, run the following command:

```
ALTER ACCOUNT
    SET CLIENT_ENCRYPTION_KEY_SIZE=256;
```

Default Value:

By default, files uploaded to internal stages are encrypted with 128-bit long AES encryption keys.

References:

1. https://docs.snowflake.com/en/sql-reference/parameters#client-encryption-key-size

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.11 Encrypt Sensitive Data at Rest Encrypt sensitive data at rest on servers, applications, and databases containing sensitive data. Storage-layer encryption, also known as server-side encryption, meets the minimum requirement of this Safeguard. Additional encryption methods may include application-layer encryption, also known as client-side encryption, where access to the data storage device(s) does not permit access to the plain-text data.		●	●
v7	14.8 Encrypt Sensitive Information at Rest Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.			●

4.3 Ensure that the DATA_RETENTION_TIME_IN_DAYS parameter is set to 90 for critical data (Manual)

Profile Applicability:

- Level 2

Description:

Snowflake Time Travel enables accessing historical data (i.e., data that has been changed or deleted) at any point within a defined period. It relies on configuring a data retention period for your critical data assets.

The `DATA_RETENTION_TIME_IN_DAYS` object parameter is used to set data retention period on the account, database, schema, or table level. When the `MIN_DATA_RETENTION_TIME_IN_DAYS` parameter is set at the account level, the effective minimum data retention period for an object is determined by `MAX(DATA_RETENTION_TIME_IN_DAYS, MIN_DATA_RETENTION_TIME_IN_DAYS).`

Rationale:

Time Travel can be used to recover critical data that was maliciously destroyed or encrypted by ransomware.

Impact:

Data retention requires additional storage which will be reflected in the monthly storage charges. For more information about storage charges, see Storage Costs for Time Travel and Fail-safe.

Audit:

Programmatically:
From a Snowsight worksheet or from the SnowSQL CLI:

1. For each table `<table_name>`, list the `DATA_RETENTION_TIME_IN_DAYS` value:

```
SHOW PARAMETERS
  LIKE 'DATA_RETENTION_TIME_IN_DAYS'
IN TABLE <table_name>;
```

2. Ensure that the parameter `value` is set to `90`.

Remediation:

An organization's compliance, legal and privacy groups may have important inputs on how long certain data should and can be retained for. For example, in the context of GDPR. It is important to take those inputs into account when data retention periods are determined for critical data.
Programmatically:
For every non-compliant table with critical data set the retention period to 90 days:

```
ALTER TABLE <table_name>
     SET DATA_RETENTION_TIME_IN_DAYS=90;
```

If all tables within a given schema or database contain critical data, the data retention period can be set on the schema or database level correspondingly.

Default Value:

The standard retention period is 1 day (24 hours) and is automatically enabled for all Snowflake accounts.

For Snowflake Standard Edition, the retention period can be set to 0 (or unset back to the default of 1 day) at the account and object level (i.e. databases, schemas, and tables).

For Snowflake Enterprise Edition (and higher):

For transient databases, schemas, and tables, the retention period can be set to 0 (or unset back to the default of 1 day). The same is also true for temporary tables.

For permanent databases, schemas, and tables, the retention period can be set to any value from 0 up to 90 days.

References:

1. https://docs.snowflake.com/en/user-guide/data-time-travel.html
2. https://docs.snowflake.com/en/user-guide/data-cdp-storage-costs.html

Additional Information:

Data retention period can only be set for permanent databases, schemas and tables. It cannot be set for transient databases, schemas and tables. It also cannot be set for temporary tables.

A threat actor with `OWNERSHIP` or `MODIFY` privilege on a database, schema, or table can override `DATA_RETENTION_TIME_IN_DAYS` parameter and effectively disable time travel, unless the `MIN_DATA_RETENTION_TIME_IN_DAYS` parameter is set at the account level.

Data retention setting is only available to Enterprise Edition or higher.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	11.1 Establish and Maintain a Data Recovery Process Establish and maintain a data recovery process. In the process, address the scope of data recovery activities, recovery prioritization, and the security of backup data. Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.	●	●	●
v8	11.2 Perform Automated Backups Perform automated backups of in-scope enterprise assets. Run backups weekly, or more frequently, based on the sensitivity of the data.	●	●	●
v7	10.2 Perform Complete System Backups Ensure that each of the organization's key systems are backed up as a complete system, through processes such as imaging, to enable the quick recovery of an entire system.	●	●	●

4.4 Ensure that the MIN_DATA_RETENTION_TIME_IN_DAYS account parameter is set to 7 or higher (Automated)

Profile Applicability:

- Level 2

Description:

The `MIN_DATA_RETENTION_TIME_IN_DAYS` account parameter can be set by users with the `ACCOUNTADMIN` role to set a minimum retention period for the account. This parameter does not alter or replace the `DATA_RETENTION_TIME_IN_DAYS` parameter value. However it may change the effective data retention time. When this parameter is set at the account level, the effective minimum data retention period for an object is determined by `MAX(DATA_RETENTION_TIME_IN_DAYS, MIN_DATA_RETENTION_TIME_IN_DAYS).`

Rationale:

Setting the `MIN_DATA_RETENTION_TIME_IN_DAYS` to 7 helps restore data-related objects (tables, schemas, and databases) that might have been accidentally or intentionally deleted.

Impact:

Data retention requires additional storage which will be reflected in the monthly storage charges. For more information about storage charges, see Storage Costs for Time Travel and Fail-safe.

Audit:

Programmatically:
In a Snowsight worksheet or from the SnowSQL CLI:

1. List the value of the parameter:

```
SHOW PARAMETERS LIKE 'MIN_DATA_RETENTION_TIME_IN_DAYS' IN ACCOUNT;
```

2. Ensure that the parameter is set to `7` or higher.

Remediation:

Programmatically:
Set the `MIN_DATA_RETENTION_TIME_IN_DAYS` on the account level to `7` or higher:

```
ALTER ACCOUNT
        SET MIN_DATA_RETENTION_TIME_IN_DAYS=7;
```

Default Value:

The default value for the `MIN_DATA_RETENTION_TIME_IN_DAYS` account parameter is `0`.

References:

1. https://docs.snowflake.com/en/user-guide/data-time-travel.html
2. https://docs.snowflake.com/en/user-guide/data-cdp-storage-costs.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.4 Enforce Data Retention Retain data according to the enterprise's data management process. Data retention must include both minimum and maximum timelines.	●	●	●
v8	11.2 Perform Automated Backups Perform automated backups of in-scope enterprise assets. Run backups weekly, or more frequently, based on the sensitivity of the data.	●	●	●
v7	10.1 Ensure Regular Automated Back Ups Ensure that all system data is automatically backed up on regular basis.	●	●	●
v7	10.2 Perform Complete System Backups Ensure that each of the organization's key systems are backed up as a complete system, through processes such as imaging, to enable the quick recovery of an entire system.	●	●	●

4.5 Ensure that the REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_CREATION account parameter is set to true (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that creating an external stage to access a private cloud storage location requires referencing a storage integration object as cloud credentials.

Rationale:

Using storage integration removes the need to supply credentials when creating external stages or when loading or unloading data. This reduces the risk of those credentials being leaked and data compromised.

Requiring a storage integration when creating a new stage reduces the risk or data exfiltration by accidentally exporting sensitive data to an external stage that does not have the appropriate network security, access control, or encryption security and is not approved by the organization's security team.

Impact:

Setting the `REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_CREATION` account level parameter to `true` can break existing manual and automated flows relying on creation of external stages not backed by a storage integration.

Audit:

Programmatically:
In the Snowsight UI or from the SnowSQL CLI:

1. List the value of the `REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_CREATION` parameter:

```
SHOW PARAMETERS LIKE 'REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_CREATION'
IN ACCOUNT;
```

2. Ensure that the parameter is set to `true`;

Remediation:

Programmatically:
In a Snowsight worksheet or from the SNOWSQL cli, run the following command to set the parameter value to `true`:

```
ALTER ACCOUNT
        SET REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_CREATION=true;
```

Note: To avoid disruption of existing workflow relying on creation of external stages not referencing a storage integration, all such workflows should be identified and migrated to creation of external stages referencing storage integrations.

Default Value:

By default, the `REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_CREATION` account level parameters is set to `false`.

References:

1. https://docs.snowflake.com/en/sql-reference/parameters#require-storage-integration-for-stage-creation
2. https://www.snowflake.com/blog/how-to-configure-a-snowflake-account-to-prevent-data-exfiltration/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.1 Establish and Maintain a Data Management Process Establish and maintain a data management process. In the process, address data sensitivity, data owner, handling of data, data retention limits, and disposal requirements, based on sensitivity and retention standards for the enterprise. Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.	●	●	●
v7	0.0 Explicitly Not Mapped Explicitly Not Mapped			

4.6 Ensure that the REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_OPERATION account parameter is set to true (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that loading data from or unloading data to a private cloud storage location requires using a named external stage that references a storage integration object.

If this parameter is not set, then users can specify the explicit cloud provider credentials directly in the `COPY` statement.

Rationale:

Using storage integration removes the need to supply credentials when loading and unloading data from external stages or when loading or unloading data to a private cloud storage location. This reduces the risk of data exfiltration by accidentally exporting sensitive data to an external stage that does not have the appropriate network security, access control, or encryption security and is not approved by the organization's security team.

Impact:

Setting the `REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_OPERATION` account level parameter to true can break existing manual and automated flows relying on loading or unloading data to external stages not backed by a storage integration.

Audit:

Programmatically:
In a Snowsight worksheet or from the SnowSQL CLI:

1. List the value of the `REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_OPERATION` parameter:

```
SHOW PARAMETERS LIKE 'REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_OPERATION'
IN ACCOUNT;
```

2. Ensure that the parameter is set to `true`.

Remediation:

Programmatically:
Set the `REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_OPERATION` on the account level to `true`:

```
ALTER ACCOUNT
   SET REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_OPERATION=true;
```

NOTE:
To avoid disruption of existing workflow relying on external stages not referencing a storage integration, all such workflows should be identified and migrated to external stages referencing storage integrations.

Default Value:

By default the `REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_OPERATION` account level parameter is set to `false`.

References:

1. https://docs.snowflake.com/en/sql-reference/parameters#require-storage-integration-for-stage-operation
2. https://www.snowflake.com/blog/how-to-configure-a-snowflake-account-to-prevent-data-exfiltration/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.1 Establish and Maintain a Data Management Process Establish and maintain a data management process. In the process, address data sensitivity, data owner, handling of data, data retention limits, and disposal requirements, based on sensitivity and retention standards for the enterprise. Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.	●	●	●
v7	0.0 Explicitly Not Mapped Explicitly Not Mapped			

4.7 Ensure that all external stages have storage integrations (Automated)

Profile Applicability:

- Level 1

Description:

External stage is a Snowflake object used for loading data from external storage locations into Snowflake tables and unloading data from Snowflake tables into external storage locations. Currently supported external storage locations are Amazon S3 buckets, Google Cloud Storage buckets and Microsoft Azure containers.

Storage integration is a Snowflake object that encapsulates external storage authentication configuration as well as an optional set of allowed or blocked storage locations. When configuring an external stage, a storage integration can be referenced in lieu of storage service credentials.

Rationale:

Using storage integration removes the need to supply credentials when creating external stages or when loading or unloading data. This reduces the risk of those credentials being leaked and data compromised.

Additionally, security administrators creating storage integration can constrain CSP storage locations allowed to be used as destinations in external stages. This further reduces the risk of data being leaked or compromised.

Impact:

None.

Audit:

Programmatically:
In the Snowsight UI or from the SnowSQL CLI:

1. List all external stages:

```
SHOW STAGES;
```

2. For each stage, ensure that if `type` is set to `EXTERNAL`, then `storage_integration` is **not** `null`.

Required privileges:
To run the query above, the caller needs the:

- `USAGE` privilege on every external stage in an account.
- `USAGE` privilege on the parenting schema of every external stage in an account.
- `USAGE` privilege on the parenting database of every external stage in an account.

Remediation:

Programmatically:

1. For each external stage, create a storage integration `<my_storage_integration>`:

```
CREATE STORAGE INTEGRATION <my_storage_integration>
  TYPE = EXTERNAL_STAGE
  STORAGE_PROVIDER = 'S3'
  ENABLED = TRUE
  STORAGE_AWS_ROLE_ARN = 'arn:aws:iam::001234567890:role/myrole';
```

2. Update the external stage `<my_external_stage>` to use the new storage integration:

```
ALTER STAGE <my_external_stage> SET STORAGE_INTEGRATION =
<my_storage_integration>;
```

Default Value:

By default, external stages may be created without storage integrations.

References:

1. https://docs.snowflake.com/en/sql-reference/sql/create-stage
2. https://docs.snowflake.com/en/sql-reference/sql/create-storage-integration
3. https://www.snowflake.com/blog/how-to-configure-a-snowflake-account-to-prevent-data-exfiltration/
4. https://docs.snowflake.com/en/sql-reference/sql/alter-stage

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.1 Establish and Maintain a Data Management Process Establish and maintain a data management process. In the process, address data sensitivity, data owner, handling of data, data retention limits, and disposal requirements, based on sensitivity and retention standards for the enterprise. Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.	●	●	●
v7	0.0 Explicitly Not Mapped Explicitly Not Mapped			

4.8 Ensure that the PREVENT_UNLOAD_TO_INLINE_URL account parameter is set to true (Automated)

Profile Applicability:

- Level 1

Description:

Prevent ad hoc data unload operations to external cloud storage by enabling the `PREVENT_UNLOAD_TO_INLINE_URL` account parameter.

Rationale:

Direct data unloading can be employed by threat actors to exfiltrate sensitive data from Snowflake to a supported external storage location of their choice. A well-intended employee with a legitimate business task can unknowingly unload data to publicly available storage locations and unintentionally leak it. Prevention of the direct data unloading reduces risk of data exfiltration and leakage.

Setting the `PREVENT_UNLOAD_TO_INLINE_URL` account parameter to `true` will prevent ad hoc data unload operations to external cloud storage locations (i.e. through `COPY INTO <location>` statements that specify the cloud storage URL and access settings directly in the statement).

Impact:

Setting the `PREVENT_UNLOAD_TO_INLINE_URL` account level parameter to true can break existing manual and automated flows relying on direct unloading data to external storage locations.

Audit:

Programmatically:
From a Snowsight worksheet or from the SnowSQL CLI:

1. List the value of the `PREVENT_UNLOAD_TO_INLINE_URL` account level parameter:

```
SHOW PARAMETERS LIKE 'PREVENT_UNLOAD_TO_INLINE_URL' IN ACCOUNT;
```

2. Ensure that `PREVENT_UNLOAD_TO_INLINE_URL` is set to `true`;

Remediation:

Programmatically:
Set the `PREVENT_UNLOAD_TO_INLINE_URL` on the account level to `true`:

```
ALTER ACCOUNT
    SET PREVENT_UNLOAD_TO_INLINE_URL=true;
```

NOTE: To avoid disruption of existing workflow relying on direct unloading data to external storage locations, all such workflows should be identified and migrated to unloading data to external stages referencing storage integrations.

Default Value:

By default the `PREVENT_UNLOAD_TO_INLINE_URL` account level parameter is set to `false`.

References:

1. https://docs.snowflake.com/en/sql-reference/sql/copy-into-location
2. https://docs.snowflake.com/en/sql-reference/parameters#prevent-unload-to-inline-url

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.1 Establish and Maintain a Data Management Process Establish and maintain a data management process. In the process, address data sensitivity, data owner, handling of data, data retention limits, and disposal requirements, based on sensitivity and retention standards for the enterprise. Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.	●	●	●
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	14.6 Protect Information through Access Control Lists Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.	●	●	●

4.9 Ensure that Tri-Secret Secure is enabled for the Snowflake account (Manual)

Profile Applicability:

- Level 2

Description:

Tri-Secret Secure is the combination of a Snowflake-maintained key and a customer-managed key in the cloud provider platform that hosts your Snowflake account to create a composite master key to protect your Snowflake data. The composite master key acts as an account master key and wraps all of the keys in the hierarchy; however, the composite master key never encrypts raw data.

Rationale:

If the customer-managed key in the composite master key hierarchy is revoked, your data can no longer be decrypted by Snowflake, providing a level of security and control above Snowflake's standard encryption.

Impact:

This feature relies on the customer managing and providing an encryption key. There is a reliability risk associated with it: If the key is lost, all data encrypted within the Snowflake account will be lost.

Audit:

Follow the instructions in the How To: Validate Tri-Secret Secure is configured for your Snowflake account successfully documentation.

Remediation:

To enable Snowflake Tri-Secret Secure for your Business Critical (or higher) account, please contact Snowflake Support.

Default Value:

By default the tri-secret secure feature is not enabled for a Snowflake account.

References:

1. https://docs.snowflake.com/en/user-guide/security-encryption-manage#tri-secret-secure
2. https://community.snowflake.com/s/article/How-to-test-Tri-Secret-Secure-is-enabled

Additional Information:

The tri-secret secure feature is currently available only to Business Critical Edition or higher.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.10 Encrypt Sensitive Data in Transit Encrypt sensitive data in transit. Example implementations can include: Transport Layer Security (TLS) and Open Secure Shell (OpenSSH).		●	●
v8	3.11 Encrypt Sensitive Data at Rest Encrypt sensitive data at rest on servers, applications, and databases containing sensitive data. Storage-layer encryption, also known as server-side encryption, meets the minimum requirement of this Safeguard. Additional encryption methods may include application-layer encryption, also known as client-side encryption, where access to the data storage device(s) does not permit access to the plain-text data.		●	●
v7	14.4 Encrypt All Sensitive Information in Transit Encrypt all sensitive information in transit.		●	●
v7	14.8 Encrypt Sensitive Information at Rest Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.			●

4.10 Ensure that data masking is enabled for sensitive data (Manual)

Profile Applicability:

- Level 2

Description:

Data masking policy is a fine-grained access control used to protect sensitive data from unauthorized access by selectively masking plain-text data in table and view columns at query time.

Rationale:

Masking policy allows for a wide range of use cases where data can be queried, aggregated and analyzed in a privacy preserving manner.

Impact:

Manual and automated workflows relying on querying unmasked data may be broken unless updated prior to application of a masking policy.

Audit:

Ensure appropriate masking policies are applied to columns with sensitive data across all tables and views in a Snowflake account.
Ensure appropriate row access policies are applied to rows with special access requirements across all tables and views in a Snowflake account.
From the UI:

1. Go to https://app.snowflake.com/ and sign into your Snowflake account.
2. On the left side navigation bar, click on *Data*.
3. Under *Data*, click on *Governance*.
4. Look for *Columns with a masking policy*. Ensure that at least one row access policy has been configured.

Programmatically:
In a Snowsight worksheet or through the SnowSQL CLI:

1. List all the configured row access policies:

```
SHOW MASKING POLICIES IN ACCOUNT;
```

2. Ensure that the query returns at least one result.

Remediation:

Identify columns with sensitive data across all account tables and views and apply appropriate masking policies following steps described in the documentation.
If columns with sensitive data are tagged appropriately, tag-based masking can be used.
Sensitive data columns can be identified and tagged with assistance of the `EXTRACT_SEMANTIC_CATEGORIES` and `ASSOCIATE_SEMANTIC_CATEGORY_TAGS` system functions. See the Data Classification documentation for details.
To create a data masking policy, follow the steps in this documentation.

Default Value:

No masking policies are applied by default in a Snowflake account.

References:

1. https://docs.snowflake.com/en/user-guide/security-column-intro
2. https://docs.snowflake.com/en/user-guide/security-column-ddm-intro
3. https://docs.snowflake.com/en/user-guide/security-column-ext-token-intro

Additional Information:

The masking policy feature is currently available only to Enterprise Edition or higher.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	0.0 Explicitly Not Mapped Explicitly Not Mapped			
v7	0.0 Explicitly Not Mapped Explicitly Not Mapped			

4.11 Ensure that row-access policies are configured for sensitive data (Manual)

Profile Applicability:

- Level 2

Description:

Row access policies are used to determine which rows to return in the query result. Row access policies can include conditions and functions in the policy expression to transform the data at query runtime when those conditions are met.

Rationale:

Row-access policy is a fine-grained access control used to protect table and view rows with special access requirements from unauthorized access at query time. It can be used to control access to certain data rows even if a user has access to query a table or view.

Impact:

Manual and automated workflows relying on having access to all rows in a table or view may be broken unless updated prior to application of a row access policy.

Audit:

Ensure appropriate row access policies are applied to rows with special access requirements across all tables and views in a Snowflake account.
From the UI:

1. Go to https://app.snowflake.com/ and sign into your Snowflake account.
2. On the left side navigation bar, click on *Data*.
3. Under *Data*, click on *Governance*.
4. Look for *Tables with row access policies*. Ensure that at least one row access policy has been configured.

Programmatically:
In a Snowsight worksheet or from the SnowSQL CLI:

1. List all the configured row access policies:

```
SHOW ROW ACCESS POLICIES IN ACCOUNT;
```

2. Ensure that the query returns at least one result.

Remediation:

Identify rows with special access requirements across all account tables and views and apply appropriate row access policies following steps described in the Using Row Access Policies documentation.

Default Value:

No row access policies are applied by default in a Snowflake account.

References:

1. https://docs.snowflake.com/en/user-guide/security-row-intro
2. https://docs.snowflake.com/en/user-guide/security-row-using

Additional Information:

The row access policy feature is currently available only to Enterprise Edition or higher.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.1 Establish and Maintain a Data Management Process Establish and maintain a data management process. In the process, address data sensitivity, data owner, handling of data, data retention limits, and disposal requirements, based on sensitivity and retention standards for the enterprise. Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.	●	●	●
v7	0.0 Explicitly Not Mapped Explicitly Not Mapped			

Appendix: Summary Table

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
1	**Identity and Access Management**		
1.1	Ensure single sign-on (SSO) is configured for your account / organization (Automated)	☐	☐
1.2	Ensure Snowflake SCIM integration is configured to automatically provision and deprovision users and groups (i.e. roles) (Automated)	☐	☐
1.3	Ensure that Snowflake password is unset for SSO users (Manual)	☐	☐
1.4	Ensure multi-factor authentication (MFA) is turned on for all human users with password-based authentication (Automated)	☐	☐
1.5	Ensure minimum password length is set to 14 characters or more (Automated)	☐	☐
1.6	Ensure that service accounts use key pair authentication (Automated)	☐	☐
1.7	Ensure authentication key pairs are rotated every 180 days (Automated)	☐	☐
1.8	Ensure that users who did not log in for 90 days are disabled (Automated)	☐	☐
1.9	Ensure that the idle session timeout is set to 15 minutes or less for users with the ACCOUNTADMIN and SECURITYADMIN roles (Automated)	☐	☐
1.10	Limit the number of users with ACCOUNTADMIN and SECURITYADMIN (Automated)	☐	☐
1.11	Ensure that all users granted the ACCOUNTADMIN role have an email address assigned (Automated)	☐	☐

CIS Benchmark Recommendation		**Set Correctly**	
		Yes	No
1.12	Ensure that no users have ACCOUNTADMIN or SECURITYADMIN as the default role (Automated)	☐	☐
1.13	Ensure that the ACCOUNTADMIN or SECURITYADMIN role is not granted to any custom role (Automated)	☐	☐
1.14	Ensure that Snowflake tasks are not owned by the ACCOUNTADMIN or SECURITYADMIN roles (Automated)	☐	☐
1.15	Ensure that Snowflake tasks do not run with the ACCOUNTADMIN or SECURITYADMIN role privileges (Automated)	☐	☐
1.16	Ensure that Snowflake stored procedures are not owned by the ACCOUNTADMIN or SECURITYADMIN roles (Automated)	☐	☐
1.17	Ensure Snowflake stored procedures do not run with ACCOUNTADMIN or SECURITYADMIN role privileges (Automated)	☐	☐
2	**Monitoring and Alerting**		
2.1	Ensure monitoring and alerting exist for ACCOUNTADMIN and SECURITYADMIN role grants (Manual)	☐	☐
2.2	Ensure monitoring and alerting exist for MANAGE GRANTS privilege grants (Manual)	☐	☐
2.3	Ensure monitoring and alerting exist for password sign-ins of SSO users (Manual)	☐	☐
2.4	Ensure monitoring and alerting exist for password sign-in without MFA (Manual)	☐	☐
2.5	Ensure monitoring and alerting exist for creation, update and deletion of security integrations (Manual)	☐	☐
2.6	Ensure monitoring and alerting exist for changes to network policies and associated objects (Manual)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
2.7	Ensure monitoring and alerting exist for SCIM token creation (Manual)	☐	☐
2.8	Ensure monitoring and alerting exists for new share exposures (Manual)	☐	☐
2.9	Ensure monitoring and alerting exists for sessions from unsupported Snowflake Connector for Python and JDBC and ODBC drivers (Manual)	☐	☐
3	**Networking**		
3.1	Ensure that an account-level network policy has been configured to only allow access from trusted IP addresses (Manual)	☐	☐
3.2	Ensure that user-level network policies have been configured for service accounts (Manual)	☐	☐
4	**Data Protection**		
4.1	Ensure yearly rekeying is enabled for a Snowflake account (Automated)	☐	☐
4.2	Ensure AES encryption key size used to encrypt files stored in internal stages is set to 256 bits (Automated)	☐	☐
4.3	Ensure that the DATA_RETENTION_TIME_IN_DAYS parameter is set to 90 for critical data (Manual)	☐	☐
4.4	Ensure that the MIN_DATA_RETENTION_TIME_IN_DAYS account parameter is set to 7 or higher (Automated)	☐	☐
4.5	Ensure that the REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_CREATION account parameter is set to true (Automated)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
4.6	Ensure that the REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_OPERATION account parameter is set to true (Automated)	☐	☐
4.7	Ensure that all external stages have storage integrations (Automated)	☐	☐
4.8	Ensure that the PREVENT_UNLOAD_TO_INLINE_URL account parameter is set to true (Automated)	☐	☐
4.9	Ensure that Tri-Secret Secure is enabled for the Snowflake account (Manual)	☐	☐
4.10	Ensure that data masking is enabled for sensitive data (Manual)	☐	☐
4.11	Ensure that row-access policies are configured for sensitive data (Manual)	☐	☐

Appendix: CIS Controls v7 IG 1 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
1.8	Ensure that users who did not log in for 90 days are disabled	☐	☐
1.9	Ensure that the idle session timeout is set to 15 minutes or less for users with the ACCOUNTADMIN and SECURITYADMIN roles	☐	☐
1.12	Ensure that no users have ACCOUNTADMIN or SECURITYADMIN as the default role	☐	☐
1.13	Ensure that the ACCOUNTADMIN or SECURITYADMIN role is not granted to any custom role	☐	☐
1.14	Ensure that Snowflake tasks are not owned by the ACCOUNTADMIN or SECURITYADMIN roles	☐	☐
1.15	Ensure that Snowflake tasks do not run with the ACCOUNTADMIN or SECURITYADMIN role privileges	☐	☐
1.16	Ensure that Snowflake stored procedures are not owned by the ACCOUNTADMIN or SECURITYADMIN roles	☐	☐
1.17	Ensure Snowflake stored procedures do not run with ACCOUNTADMIN or SECURITYADMIN role privileges	☐	☐
3.1	Ensure that an account-level network policy has been configured to only allow access from trusted IP addresses	☐	☐
3.2	Ensure that user-level network policies have been configured for service accounts	☐	☐
4.3	Ensure that the DATA_RETENTION_TIME_IN_DAYS parameter is set to 90 for critical data	☐	☐
4.4	Ensure that the MIN_DATA_RETENTION_TIME_IN_DAYS account parameter is set to 7 or higher	☐	☐
4.8	Ensure that the PREVENT_UNLOAD_TO_INLINE_URL account parameter is set to true	☐	☐

Appendix: CIS Controls v7 IG 2 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
1.1	Ensure single sign-on (SSO) is configured for your account / organization	☐	☐
1.2	Ensure Snowflake SCIM integration is configured to automatically provision and deprovision users and groups (i.e. roles)	☐	☐
1.4	Ensure multi-factor authentication (MFA) is turned on for all human users with password-based authentication	☐	☐
1.5	Ensure minimum password length is set to 14 characters or more	☐	☐
1.8	Ensure that users who did not log in for 90 days are disabled	☐	☐
1.9	Ensure that the idle session timeout is set to 15 minutes or less for users with the ACCOUNTADMIN and SECURITYADMIN roles	☐	☐
1.12	Ensure that no users have ACCOUNTADMIN or SECURITYADMIN as the default role	☐	☐
1.13	Ensure that the ACCOUNTADMIN or SECURITYADMIN role is not granted to any custom role	☐	☐
1.14	Ensure that Snowflake tasks are not owned by the ACCOUNTADMIN or SECURITYADMIN roles	☐	☐
1.15	Ensure that Snowflake tasks do not run with the ACCOUNTADMIN or SECURITYADMIN role privileges	☐	☐
1.16	Ensure that Snowflake stored procedures are not owned by the ACCOUNTADMIN or SECURITYADMIN roles	☐	☐
1.17	Ensure Snowflake stored procedures do not run with ACCOUNTADMIN or SECURITYADMIN role privileges	☐	☐
2.1	Ensure monitoring and alerting exist for ACCOUNTADMIN and SECURITYADMIN role grants	☐	☐
2.2	Ensure monitoring and alerting exist for MANAGE GRANTS privilege grants	☐	☐

Recommendation		Set Correctly	
		Yes	No
2.3	Ensure monitoring and alerting exist for password sign-ins of SSO users	☐	☐
2.4	Ensure monitoring and alerting exist for password sign-in without MFA	☐	☐
2.5	Ensure monitoring and alerting exist for creation, update and deletion of security integrations	☐	☐
2.6	Ensure monitoring and alerting exist for changes to network policies and associated objects	☐	☐
2.7	Ensure monitoring and alerting exist for SCIM token creation	☐	☐
2.8	Ensure monitoring and alerting exists for new share exposures	☐	☐
2.9	Ensure monitoring and alerting exists for sessions from unsupported Snowflake Connector for Python and JDBC and ODBC drivers	☐	☐
3.1	Ensure that an account-level network policy has been configured to only allow access from trusted IP addresses	☐	☐
3.2	Ensure that user-level network policies have been configured for service accounts	☐	☐
4.3	Ensure that the DATA_RETENTION_TIME_IN_DAYS parameter is set to 90 for critical data	☐	☐
4.4	Ensure that the MIN_DATA_RETENTION_TIME_IN_DAYS account parameter is set to 7 or higher	☐	☐
4.8	Ensure that the PREVENT_UNLOAD_TO_INLINE_URL account parameter is set to true	☐	☐
4.9	Ensure that Tri-Secret Secure is enabled for the Snowflake account	☐	☐

Appendix: CIS Controls v7 IG 3 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
1.1	Ensure single sign-on (SSO) is configured for your account / organization	☐	☐
1.2	Ensure Snowflake SCIM integration is configured to automatically provision and deprovision users and groups (i.e. roles)	☐	☐
1.4	Ensure multi-factor authentication (MFA) is turned on for all human users with password-based authentication	☐	☐
1.5	Ensure minimum password length is set to 14 characters or more	☐	☐
1.8	Ensure that users who did not log in for 90 days are disabled	☐	☐
1.9	Ensure that the idle session timeout is set to 15 minutes or less for users with the ACCOUNTADMIN and SECURITYADMIN roles	☐	☐
1.12	Ensure that no users have ACCOUNTADMIN or SECURITYADMIN as the default role	☐	☐
1.13	Ensure that the ACCOUNTADMIN or SECURITYADMIN role is not granted to any custom role	☐	☐
1.14	Ensure that Snowflake tasks are not owned by the ACCOUNTADMIN or SECURITYADMIN roles	☐	☐
1.15	Ensure that Snowflake tasks do not run with the ACCOUNTADMIN or SECURITYADMIN role privileges	☐	☐
1.16	Ensure that Snowflake stored procedures are not owned by the ACCOUNTADMIN or SECURITYADMIN roles	☐	☐
1.17	Ensure Snowflake stored procedures do not run with ACCOUNTADMIN or SECURITYADMIN role privileges	☐	☐
2.1	Ensure monitoring and alerting exist for ACCOUNTADMIN and SECURITYADMIN role grants	☐	☐
2.2	Ensure monitoring and alerting exist for MANAGE GRANTS privilege grants	☐	☐

Recommendation		Set Correctly	
		Yes	No
2.3	Ensure monitoring and alerting exist for password sign-ins of SSO users	☐	☐
2.4	Ensure monitoring and alerting exist for password sign-in without MFA	☐	☐
2.5	Ensure monitoring and alerting exist for creation, update and deletion of security integrations	☐	☐
2.6	Ensure monitoring and alerting exist for changes to network policies and associated objects	☐	☐
2.7	Ensure monitoring and alerting exist for SCIM token creation	☐	☐
2.8	Ensure monitoring and alerting exists for new share exposures	☐	☐
2.9	Ensure monitoring and alerting exists for sessions from unsupported Snowflake Connector for Python and JDBC and ODBC drivers	☐	☐
3.1	Ensure that an account-level network policy has been configured to only allow access from trusted IP addresses	☐	☐
3.2	Ensure that user-level network policies have been configured for service accounts	☐	☐
4.1	Ensure yearly rekeying is enabled for a Snowflake account	☐	☐
4.2	Ensure AES encryption key size used to encrypt files stored in internal stages is set to 256 bits	☐	☐
4.3	Ensure that the DATA_RETENTION_TIME_IN_DAYS parameter is set to 90 for critical data	☐	☐
4.4	Ensure that the MIN_DATA_RETENTION_TIME_IN_DAYS account parameter is set to 7 or higher	☐	☐
4.8	Ensure that the PREVENT_UNLOAD_TO_INLINE_URL account parameter is set to true	☐	☐
4.9	Ensure that Tri-Secret Secure is enabled for the Snowflake account	☐	☐

Appendix: CIS Controls v8 IG 1 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
1.4	Ensure multi-factor authentication (MFA) is turned on for all human users with password-based authentication	☐	☐
1.5	Ensure minimum password length is set to 14 characters or more	☐	☐
1.8	Ensure that users who did not log in for 90 days are disabled	☐	☐
1.9	Ensure that the idle session timeout is set to 15 minutes or less for users with the ACCOUNTADMIN and SECURITYADMIN roles	☐	☐
1.12	Ensure that no users have ACCOUNTADMIN or SECURITYADMIN as the default role	☐	☐
1.13	Ensure that the ACCOUNTADMIN or SECURITYADMIN role is not granted to any custom role	☐	☐
1.14	Ensure that Snowflake tasks are not owned by the ACCOUNTADMIN or SECURITYADMIN roles	☐	☐
1.15	Ensure that Snowflake tasks do not run with the ACCOUNTADMIN or SECURITYADMIN role privileges	☐	☐
1.16	Ensure that Snowflake stored procedures are not owned by the ACCOUNTADMIN or SECURITYADMIN roles	☐	☐
1.17	Ensure Snowflake stored procedures do not run with ACCOUNTADMIN or SECURITYADMIN role privileges	☐	☐
3.1	Ensure that an account-level network policy has been configured to only allow access from trusted IP addresses	☐	☐
3.2	Ensure that user-level network policies have been configured for service accounts	☐	☐
4.3	Ensure that the DATA_RETENTION_TIME_IN_DAYS parameter is set to 90 for critical data	☐	☐
4.4	Ensure that the MIN_DATA_RETENTION_TIME_IN_DAYS account parameter is set to 7 or higher	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.5	Ensure that the REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_CREATION account parameter is set to true	☐	☐
4.6	Ensure that the REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_OPERATION account parameter is set to true	☐	☐
4.7	Ensure that all external stages have storage integrations	☐	☐
4.8	Ensure that the PREVENT_UNLOAD_TO_INLINE_URL account parameter is set to true	☐	☐
4.11	Ensure that row-access policies are configured for sensitive data	☐	☐

Appendix: CIS Controls v8 IG 2 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
1.1	Ensure single sign-on (SSO) is configured for your account / organization	☐	☐
1.2	Ensure Snowflake SCIM integration is configured to automatically provision and deprovision users and groups (i.e. roles)	☐	☐
1.4	Ensure multi-factor authentication (MFA) is turned on for all human users with password-based authentication	☐	☐
1.5	Ensure minimum password length is set to 14 characters or more	☐	☐
1.8	Ensure that users who did not log in for 90 days are disabled	☐	☐
1.9	Ensure that the idle session timeout is set to 15 minutes or less for users with the ACCOUNTADMIN and SECURITYADMIN roles	☐	☐
1.12	Ensure that no users have ACCOUNTADMIN or SECURITYADMIN as the default role	☐	☐
1.13	Ensure that the ACCOUNTADMIN or SECURITYADMIN role is not granted to any custom role	☐	☐
1.14	Ensure that Snowflake tasks are not owned by the ACCOUNTADMIN or SECURITYADMIN roles	☐	☐
1.15	Ensure that Snowflake tasks do not run with the ACCOUNTADMIN or SECURITYADMIN role privileges	☐	☐
1.16	Ensure that Snowflake stored procedures are not owned by the ACCOUNTADMIN or SECURITYADMIN roles	☐	☐
1.17	Ensure Snowflake stored procedures do not run with ACCOUNTADMIN or SECURITYADMIN role privileges	☐	☐
2.1	Ensure monitoring and alerting exist for ACCOUNTADMIN and SECURITYADMIN role grants	☐	☐
2.2	Ensure monitoring and alerting exist for MANAGE GRANTS privilege grants	☐	☐

Recommendation		Set Correctly	
		Yes	No
2.3	Ensure monitoring and alerting exist for password sign-ins of SSO users	☐	☐
2.4	Ensure monitoring and alerting exist for password sign-in without MFA	☐	☐
2.5	Ensure monitoring and alerting exist for creation, update and deletion of security integrations	☐	☐
2.6	Ensure monitoring and alerting exist for changes to network policies and associated objects	☐	☐
2.7	Ensure monitoring and alerting exist for SCIM token creation	☐	☐
2.8	Ensure monitoring and alerting exists for new share exposures	☐	☐
2.9	Ensure monitoring and alerting exists for sessions from unsupported Snowflake Connector for Python and JDBC and ODBC drivers	☐	☐
3.1	Ensure that an account-level network policy has been configured to only allow access from trusted IP addresses	☐	☐
3.2	Ensure that user-level network policies have been configured for service accounts	☐	☐
4.1	Ensure yearly rekeying is enabled for a Snowflake account	☐	☐
4.2	Ensure AES encryption key size used to encrypt files stored in internal stages is set to 256 bits	☐	☐
4.3	Ensure that the DATA_RETENTION_TIME_IN_DAYS parameter is set to 90 for critical data	☐	☐
4.4	Ensure that the MIN_DATA_RETENTION_TIME_IN_DAYS account parameter is set to 7 or higher	☐	☐
4.5	Ensure that the REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_CREATION account parameter is set to true	☐	☐
4.6	Ensure that the REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_OPERATION account parameter is set to true	☐	☐
4.7	Ensure that all external stages have storage integrations	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.8	Ensure that the PREVENT_UNLOAD_TO_INLINE_URL account parameter is set to true	☐	☐
4.9	Ensure that Tri-Secret Secure is enabled for the Snowflake account	☐	☐
4.11	Ensure that row-access policies are configured for sensitive data	☐	☐

Appendix: CIS Controls v8 IG 3 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
1.1	Ensure single sign-on (SSO) is configured for your account / organization	☐	☐
1.2	Ensure Snowflake SCIM integration is configured to automatically provision and deprovision users and groups (i.e. roles)	☐	☐
1.4	Ensure multi-factor authentication (MFA) is turned on for all human users with password-based authentication	☐	☐
1.5	Ensure minimum password length is set to 14 characters or more	☐	☐
1.8	Ensure that users who did not log in for 90 days are disabled	☐	☐
1.9	Ensure that the idle session timeout is set to 15 minutes or less for users with the ACCOUNTADMIN and SECURITYADMIN roles	☐	☐
1.12	Ensure that no users have ACCOUNTADMIN or SECURITYADMIN as the default role	☐	☐
1.13	Ensure that the ACCOUNTADMIN or SECURITYADMIN role is not granted to any custom role	☐	☐
1.14	Ensure that Snowflake tasks are not owned by the ACCOUNTADMIN or SECURITYADMIN roles	☐	☐
1.15	Ensure that Snowflake tasks do not run with the ACCOUNTADMIN or SECURITYADMIN role privileges	☐	☐
1.16	Ensure that Snowflake stored procedures are not owned by the ACCOUNTADMIN or SECURITYADMIN roles	☐	☐
1.17	Ensure Snowflake stored procedures do not run with ACCOUNTADMIN or SECURITYADMIN role privileges	☐	☐
2.1	Ensure monitoring and alerting exist for ACCOUNTADMIN and SECURITYADMIN role grants	☐	☐
2.2	Ensure monitoring and alerting exist for MANAGE GRANTS privilege grants	☐	☐

Recommendation		Set Correctly	
		Yes	No
2.3	Ensure monitoring and alerting exist for password sign-ins of SSO users	☐	☐
2.4	Ensure monitoring and alerting exist for password sign-in without MFA	☐	☐
2.5	Ensure monitoring and alerting exist for creation, update and deletion of security integrations	☐	☐
2.6	Ensure monitoring and alerting exist for changes to network policies and associated objects	☐	☐
2.7	Ensure monitoring and alerting exist for SCIM token creation	☐	☐
2.8	Ensure monitoring and alerting exists for new share exposures	☐	☐
2.9	Ensure monitoring and alerting exists for sessions from unsupported Snowflake Connector for Python and JDBC and ODBC drivers	☐	☐
3.1	Ensure that an account-level network policy has been configured to only allow access from trusted IP addresses	☐	☐
3.2	Ensure that user-level network policies have been configured for service accounts	☐	☐
4.1	Ensure yearly rekeying is enabled for a Snowflake account	☐	☐
4.2	Ensure AES encryption key size used to encrypt files stored in internal stages is set to 256 bits	☐	☐
4.3	Ensure that the DATA_RETENTION_TIME_IN_DAYS parameter is set to 90 for critical data	☐	☐
4.4	Ensure that the MIN_DATA_RETENTION_TIME_IN_DAYS account parameter is set to 7 or higher	☐	☐
4.5	Ensure that the REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_CREATION account parameter is set to true	☐	☐
4.6	Ensure that the REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_OPERATION account parameter is set to true	☐	☐
4.7	Ensure that all external stages have storage integrations	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.8	Ensure that the PREVENT_UNLOAD_TO_INLINE_URL account parameter is set to true	☐	☐
4.9	Ensure that Tri-Secret Secure is enabled for the Snowflake account	☐	☐
4.11	Ensure that row-access policies are configured for sensitive data	☐	☐

Appendix: Change History

Date	Version	Changes for this version
10/27/2023	1.0.0	Initial Release