Vignette 4: Horizon によるガバナンス (Governance with Horizon)

概要

この Vignette では、Snowflake Horizon 内の強力なガバナンス機能のいくつかを探ります。ロールベースのアクセス制御 (RBAC) の確認から始め、自動データ分類、列レベルのセキュリティのためのタグベースのマスキングポリシー、行アクセスポリシー、データ品質監視、そして最後に Trust Center によるアカウント全体のセキュリティ監視などの機能を深く掘り下げます。

学習内容

- Snowflake におけるロールベースのアクセス制御 (RBAC) の基礎。
- 機密データを自動的に分類してタグ付けする方法。
- 動的データマスキングを使用して列レベルのセキュリティを実装する方法。
- 行アクセスポリシーを使用して行レベルのセキュリティを実装する方法。
- データメトリック関数を使用してデータ品質を監視する方法。
- Trust Center を使用してアカウントのセキュリティを監視する方法。

構築するもの

- カスタムの特権ロール。
- PII を自動タグ付けするためのデータ分類プロファイル。
- 文字列および日付列用のタグベースのマスキングポリシー。
- 国ごとにデータの可視性を制限する行アクセスポリシー。
- データの整合性をチェックするためのカスタムデータメトリック関数。

SQLコードを取得してSQLファイルに貼り付ける

この ファイル から SQL をコピーして、新しい SQL ファイルに貼り付け、Snowflake でフォローしてください。

注: SQL ファイルの最後に到達したら、ステップ 29 - アプリとコラボレーション に進むことができます。

ロールとアクセス制御 (Roles and Access Control)

概要

Snowflake のセキュリティモデルは、ロールベースのアクセス制御 (RBAC) と任意アクセス制御 (DAC) のフレームワーク上に構築されています。アクセス権限はロールに割り当てられ、ロールはユーザーに割り当てられます。これにより、オブジェクトを保護するための強力で柔軟な階層が作成されます。

> **アクセス制御の概要**: セキュリティ保護可能なオブジェクト、ロール、権限、ユーザーなど、Snowflake のアクセス制御の主要な概念について詳しく学びます。

ステップ 1 - コンテキストの設定と既存ロールの表示

まず、この演習のコンテキストを設定し、アカウントに既に存在するロールを表示しましょう。

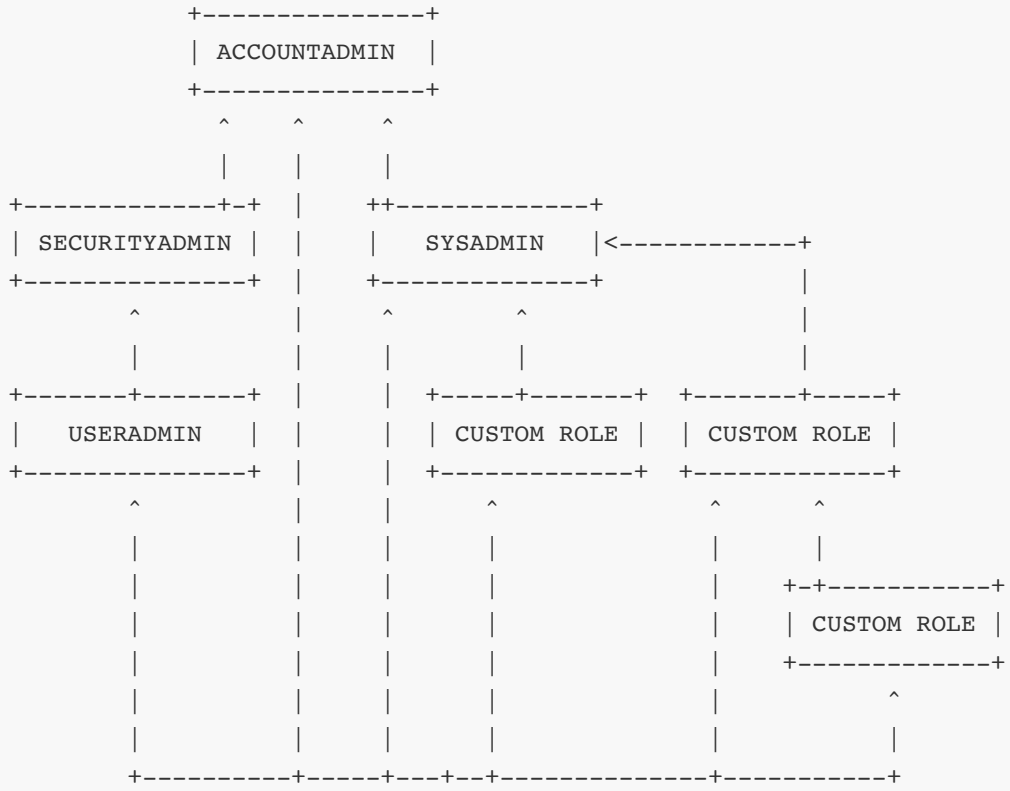
```
USE ROLE useradmin;
USE DATABASE tb_101;
USE WAREHOUSE tb_dev_wh;

SHOW ROLES;
```

ステップ 2 - カスタムロールの作成

次に、カスタムの `tb_data_steward` ロールを作成します。このロールは、顧客データの管理と保護を担当します。

```
CREATE OR REPLACE ROLE tb_data_steward
    COMMENT = 'Custom Role';
```

システムロールとカスタムロールの一般的な階層は、次のようになります:



```
                                    |
                               +----+-----+
                               |  PUBLIC  |
                               +----------+
```

Snowflake システム定義ロールの定義:

- **ORGADMIN**: 組織レベルで操作を管理するロール。
- **ACCOUNTADMIN**: システム内の最上位ロールであり、アカウント内の限られた/制御された数のユーザーにのみ付与する必要があります。
- **SECURITYADMIN**: オブジェクトの付与をグローバルに管理し、ユーザーとロールを作成、監視、管理できるロール。
- **USERADMIN**: ユーザーとロールの管理のみに特化したロール。
- **SYSADMIN**: アカウント内にウェアハウスとデータベースを作成する権限を持つロール。
- **PUBLIC**: すべてのユーザーとロールに自動的に付与される疑似ロール。セキュリティ保護可能なオブジェクトを所有でき、所有するものはすべて、アカウント内の他のすべてのユーザーとロールが利用できるようになります。

ステップ 3 - カスタムロールへの権限の付与

権限を付与しないと、ロールでできることはあまりありません。`securityadmin` ロールに切り替えて、新しい `tb_data_steward` ロールに、ウェアハウスの使用とデータベーススキーマおよびテーブルへのアクセスに必要な権限を付与しましょう。

```
USE ROLE securityadmin;

-- ウェアハウスの使用権限を付与
GRANT OPERATE, USAGE ON WAREHOUSE tb_dev_wh TO ROLE tb_data_steward;

-- データベースとスキーマの使用権限を付与
GRANT USAGE ON DATABASE tb_101 TO ROLE tb_data_steward;
GRANT USAGE ON ALL SCHEMAS IN DATABASE tb_101 TO ROLE tb_data_steward;

-- テーブルレベルの権限を付与
GRANT SELECT ON ALL TABLES IN SCHEMA raw_customer TO ROLE tb_data_steward;
GRANT ALL ON SCHEMA governance TO ROLE tb_data_steward;
GRANT ALL ON ALL TABLES IN SCHEMA governance TO ROLE tb_data_steward;
```

ステップ 4 - 新しいロールの付与と使用

最後に、新しいロールを自分のユーザーに付与します。その後、`tb_data_steward` ロールに切り替えてクエリを実行し、アクセスできるデータを確認できます。

```
-- 自分のユーザーにロールを付与
SET my_user = CURRENT_USER();
GRANT ROLE tb_data_steward TO USER IDENTIFIER($my_user);

-- 新しいロールに切り替え
USE ROLE tb_data_steward;

-- テストクエリを実行
SELECT TOP 100 * FROM raw_customer.customer_loyalty;
```

クエリ結果を見ると、このテーブルには多くの個人識別情報 (PII) が含まれていることが明らかです。次のセクションでは、これらを保護する方法を学びます。

分類と自動タグ付け (Classification and Auto Tagging)

概要

データガバナンスの重要な最初のステップは、機密データの特定と分類です。Snowflake Horizon の自動タグ付け機能は、スキーマ内の列を監視することで、機密情報を自動的に検出できます。その後、これらのタグを使用してセキュリティポリシーを適用できます。

> **自動分類**: スケジュールに基づいて機密データを自動的に分類し、大規模なガバナンスを簡素化する方法について学びます。

ステップ 1 - PII タグの作成と権限の付与

`accountadmin` ロールを使用して、`governance` スキーマに `pii` タグを作成します。また、分類を実行するために必要な権限を `tb_data_steward` ロールに付与します。

```
USE ROLE accountadmin;

CREATE OR REPLACE TAG governance.pii;
GRANT APPLY TAG ON ACCOUNT TO ROLE tb_data_steward;

GRANT EXECUTE AUTO CLASSIFICATION ON SCHEMA raw_customer TO ROLE tb_data_steward;
GRANT DATABASE ROLE SNOWFLAKE.CLASSIFICATION_ADMIN TO ROLE tb_data_steward;
GRANT CREATE SNOWFLAKE.DATA_PRIVACY.CLASSIFICATION_PROFILE ON SCHEMA governance TO ROLE
tb_data_steward;
```

ステップ 2 - 分類プロファイルの作成

今度は `tb_data_steward` として、分類プロファイルを作成します。このプロファイルは、自動タグ付けの動作を定義します。

```
USE ROLE tb_data_steward;

CREATE OR REPLACE SNOWFLAKE.DATA_PRIVACY.CLASSIFICATION_PROFILE
  governance.tb_classification_profile(
    {
      'minimum_object_age_for_classification_days': 0,
      'maximum_classification_validity_days': 30,
      'auto_tag': true
    });
```

ステップ 3 - セマンティックカテゴリの PII タグへのマッピング

次に、`SEMANTIC_CATEGORY` が `NAME`、`PHONE_NUMBER`、`EMAIL` などの一般的な PII タイプと一致するすべての列に `governance.pii` タグを適用するように分類プロファイルに指示するマッピングを定義します。

```
CALL governance.tb_classification_profile!SET_TAG_MAP(
  {'column_tag_map':[
    {
      'tag_name':'tb_101.governance.pii',
      'tag_value':'pii',
      'semantic_categories':['NAME', 'PHONE_NUMBER', 'POSTAL_CODE', 'DATE_OF_BIRTH',
'CITY', 'EMAIL']
    }]});
```

ステップ 4 - 分類の実行と結果の表示

`customer_loyalty` テーブルで分類プロセスを手動でトリガーしてみましょう。その後、`INFORMATION_SCHEMA` をクエリして、自動的に適用されたタグを確認できます。

```
-- 分類をトリガー
CALL SYSTEM$CLASSIFY('tb_101.raw_customer.customer_loyalty',
'tb_101.governance.tb_classification_profile');

-- 適用されたタグを表示
SELECT
    column_name,
    tag_database,
    tag_schema,
    tag_name,
    tag_value,
    apply_method
FROM TABLE(INFORMATION_SCHEMA.TAG_REFERENCES_ALL_COLUMNS('raw_customer.customer_loyalty',
'table'));
```

PII として識別された列に、カスタムの `governance.pii` タグが適用されていることに注目してください。

マスキングポリシー (Masking Policies)

概要

機密列にタグ付けされたので、動的データマスキングを使用してそれらを保護できます。マスキングポリシーは、クエリ時にユーザーに元のデータを表示するか、マスクされたバージョンを表示するかを決定するスキーマレベルのオブジェクトです。これらのポリシーを `pii` タグに直接適用できます。

> [列レベルセキュリティ](): 列レベルセキュリティには、機密データを保護するための動的データマスキングと外部トークン化が含まれます。

ステップ 1 - マスキングポリシーの作成

2 つのポリシーを作成します。1 つは文字列データをマスクするため、もう 1 つは日付データをマスクするためです。ロジックは単純です。ユーザーのロールが特権ロール（つまり、`ACCOUNTADMIN` または `TB_ADMIN` ではない）でない場合は、マスクされた値を返します。それ以外の場合は、元の値を返します。

```
-- 機密文字列データ用のマスキングポリシーを作成
CREATE OR REPLACE MASKING POLICY governance.mask_string_pii AS (original_value STRING)
RETURNS STRING ->
  CASE WHEN
    CURRENT_ROLE() NOT IN ('ACCOUNTADMIN', 'TB_ADMIN')
    THEN '****MASKED****'
    ELSE original_value
  END;

-- 機密日付データ用のマスキングポリシーを作成
CREATE OR REPLACE MASKING POLICY governance.mask_date_pii AS (original_value DATE)
RETURNS DATE ->
  CASE WHEN
    CURRENT_ROLE() NOT IN ('ACCOUNTADMIN', 'TB_ADMIN')
    THEN DATE_TRUNC('year', original_value)
    ELSE original_value
  END;
```

ステップ 2 - マスキングポリシーのタグへの適用

タグベースのガバナンスの威力は、ポリシーをタグに一度適用することにあります。このアクションにより、そのタグを持つすべての列が、現在および将来にわたって自動的に保護されます。

```
ALTER TAG governance.pii SET
    MASKING POLICY governance.mask_string_pii,
    MASKING POLICY governance.mask_date_pii;
```

ステップ 3 - ポリシーのテスト

作業内容をテストしてみましょう。まず、非特権ロール `public` に切り替えてテーブルをクエリします。PII 列はマスクされているはずです。

```
USE ROLE public;
SELECT TOP 100 * FROM raw_customer.customer_loyalty;
```

次に、特権ロール `tb_admin` に切り替えます。データは完全に表示されるはずです。

```
USE ROLE tb_admin;
SELECT TOP 100 * FROM raw_customer.customer_loyalty;
```

行アクセスポリシー (Row Access Policies)

概要

列のマスキングに加えて、Snowflake では、行アクセスポリシーを使用してユーザーに表示される行をフィルタリングできます。ポリシーは、多くの場合ユーザーのロールやその他のセッション属性に基づいて、定義したルールに対して各行を評価します。

> 行レベルセキュリティ: 行アクセスポリシーは、クエリ結果に表示される行を決定し、きめ細かいアクセス制御を可能にします。

ステップ 1 - ポリシーマッピングテーブルの作成

行アクセスポリシーの一般的なパターンは、どのロールがどのデータを表示できるかを定義するマッピングテーブルを使用することです。ロールを、表示が許可されている `country` 値にマッピングするテーブルを作成します。

```
USE ROLE tb_data_steward;

CREATE OR REPLACE TABLE governance.row_policy_map
    (role STRING, country_permission STRING);

-- tb_data_engineer ロールを 'United States' のデータのみ表示するようにマッピング
INSERT INTO governance.row_policy_map
    VALUES('tb_data_engineer', 'United States');
```

ステップ 2 - 行アクセスポリシーの作成

次に、ポリシー自体を作成します。このポリシーは、ユーザーのロールが管理者ロールである場合、またはユーザーのロールがマッピングテーブルに存在し、現在の行の `country` 値と一致する場合に `TRUE`（行の表示を許可）を返します。

```
CREATE OR REPLACE ROW ACCESS POLICY governance.customer_loyalty_policy
    AS (country STRING) RETURNS BOOLEAN ->
        CURRENT_ROLE() IN ('ACCOUNTADMIN', 'SYSADMIN')
        OR EXISTS
            (
            SELECT 1 FROM governance.row_policy_map rp
            WHERE
                UPPER(rp.role) = CURRENT_ROLE()
                AND rp.country_permission = country
            );
```

ステップ 3 - ポリシーの適用とテスト

`customer_loyalty` テーブルの `country` 列にポリシーを適用します。次に、`tb_data_engineer` ロールに切り替えてテーブルをクエリします。

```
-- ポリシーを適用
ALTER TABLE raw_customer.customer_loyalty
    ADD ROW ACCESS POLICY governance.customer_loyalty_policy ON (country);

-- ロールを切り替えてポリシーをテスト
USE ROLE tb_data_engineer;

-- テーブルをクエリ
SELECT TOP 100 * FROM raw_customer.customer_loyalty;
```

結果セットには、`country` が 'United States' である行のみが含まれるはずです。

データメトリック関数 (Data Metric Functions)

概要

データガバナンスはセキュリティだけではありません。信頼性と信頼性も重要です。Snowflake は、データメトリック関数 (DMF) を使用してデータの整合性を維持するのに役立ちます。システム定義の DMF を使用するか、独自に作成して、テーブルに対して自動品質チェックを実行できます。

> **データ品質監視**: 組み込みおよびカスタムのデータメトリック関数を使用して、データの一貫性と信頼性を確保する方法について学びます。

ステップ 1 - システム DMF の使用

Snowflake の組み込み DMF のいくつかを使用して、`order_header` テーブルの品質を確認してみましょう。

```
USE ROLE tb_data_steward;

-- これは、顧客 ID が null の割合を返します。
SELECT SNOWFLAKE.CORE.NULL_PERCENT(SELECT customer_id FROM raw_pos.order_header);

-- DUPLICATE_COUNT を使用して、重複する注文 ID を確認できます。
SELECT SNOWFLAKE.CORE.DUPLICATE_COUNT(SELECT order_id FROM raw_pos.order_header);

-- すべての注文の平均注文合計金額。
SELECT SNOWFLAKE.CORE.AVG(SELECT order_total FROM raw_pos.order_header);
```

ステップ 2 - カスタム DMF の作成

特定のビジネスロジック用にカスタム DMF を作成することもできます。`order_total` が `unit_price * quantity` と等しくない注文をチェックするものを作成しましょう。

```
CREATE OR REPLACE DATA METRIC FUNCTION governance.invalid_order_total_count(
    order_prices_t table(
        order_total NUMBER,
        unit_price NUMBER,
        quantity INTEGER
    )
)
RETURNS NUMBER
AS
'SELECT COUNT(*)
 FROM order_prices_t
 WHERE order_total != unit_price * quantity';
```

ステップ 3 - DMF のテストとスケジュール設定

DMF をテストするために、不良レコードを挿入しましょう。その後、関数を呼び出してエラーが検出されるかどうかを確認します。挿入するレコードは、単価 5 ドルの商品を 2 つ注文し、正しい合計 10 ドルではなく合計価格が 5 ドルになっているものです。

```
-- 合計価格が正しくないレコードを挿入
INSERT INTO raw_pos.order_detail
SELECT 904745311, 459520442, 52, null, 0, 2, 5.0, 5.0, null;

-- 注文詳細テーブルでカスタム DMF を呼び出す
SELECT governance.invalid_order_total_count(
    SELECT price, unit_price, quantity FROM raw_pos.order_detail
) AS num_orders_with_incorrect_price;
```

このチェックを自動化するために、DMF をテーブルに関連付け、データが変更されるたびに自動的に実行されるようにスケジュールを設定してから、`order_detail` テーブルに追加できます。

```
ALTER TABLE raw_pos.order_detail
    SET DATA_METRIC_SCHEDULE = 'TRIGGER_ON_CHANGES';

ALTER TABLE raw_pos.order_detail
    ADD DATA METRIC FUNCTION governance.invalid_order_total_count
    ON (price, unit_price, quantity);
```

Trust Center (トラストセンター)

概要

Trust Center は、Snowflake アカウント全体のセキュリティリスクを監視するための一元化されたダッシュボードを提供します。スケジュールされたスキャナーを使用して、多要素認証 (MFA) の欠落、過剰な特権を持つロール、非アクティブなユーザーなどの問題をチェックし、推奨されるアクションを提供します。

> **Trust Center の概要**: Trust Center は、アカウントのセキュリティリスクを評価および監視するための自動チェックを可能にします。

ステップ 1 - 権限の付与と Trust Center への移動

まず、`ACCOUNTADMIN` は、`TRUST_CENTER_ADMIN` アプリケーションロールをユーザーまたはロールに付与する必要があります。これを `tb_admin` ロールに付与します。

```
USE ROLE accountadmin;
GRANT APPLICATION ROLE SNOWFLAKE.TRUST_CENTER_ADMIN TO ROLE tb_admin;
USE ROLE tb_admin;
```

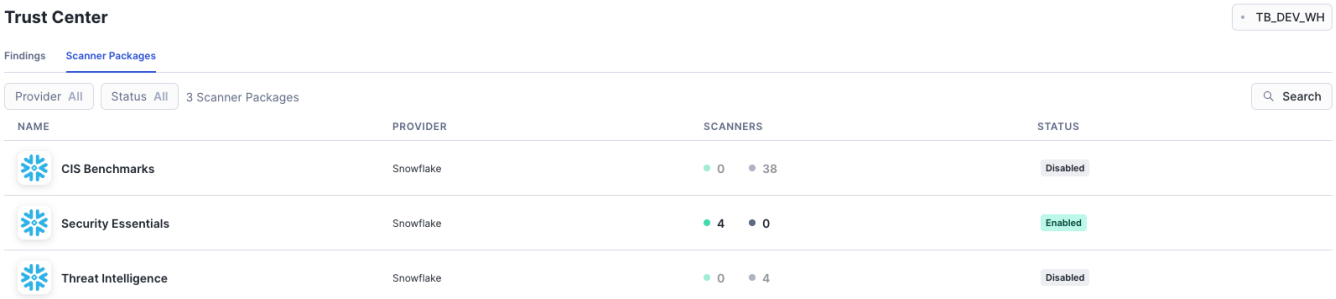
次に、Snowsight UI で Trust Center に移動します:

1. 左側のナビゲーションバーにある **Monitoring** タブをクリックします。
2. **Trust Center** をクリックします。

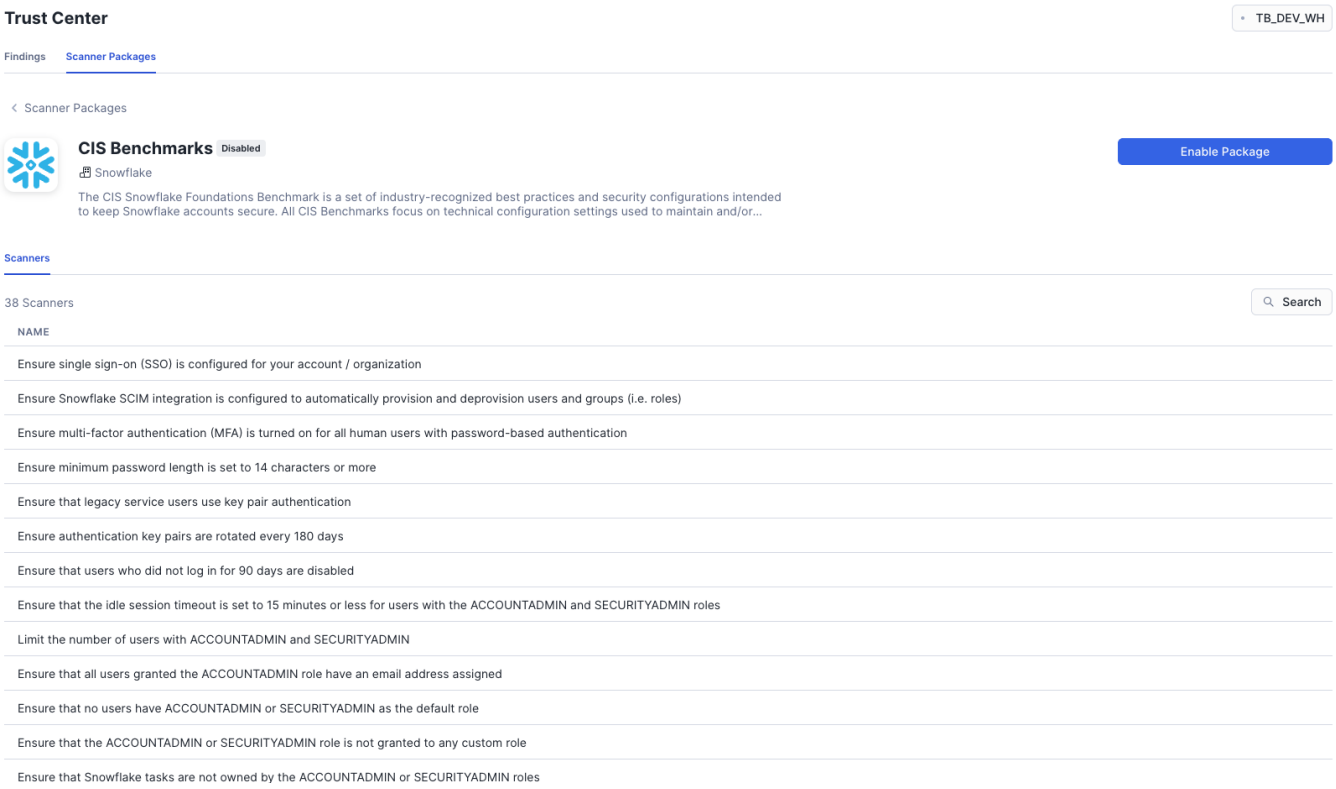
ステップ 2 - スキャナーパッケージの有効化

デフォルトでは、ほとんどのスキャナーパッケージは無効になっています。アカウントのセキュリティ体制を包括的に把握するために、これらを有効にしましょう。

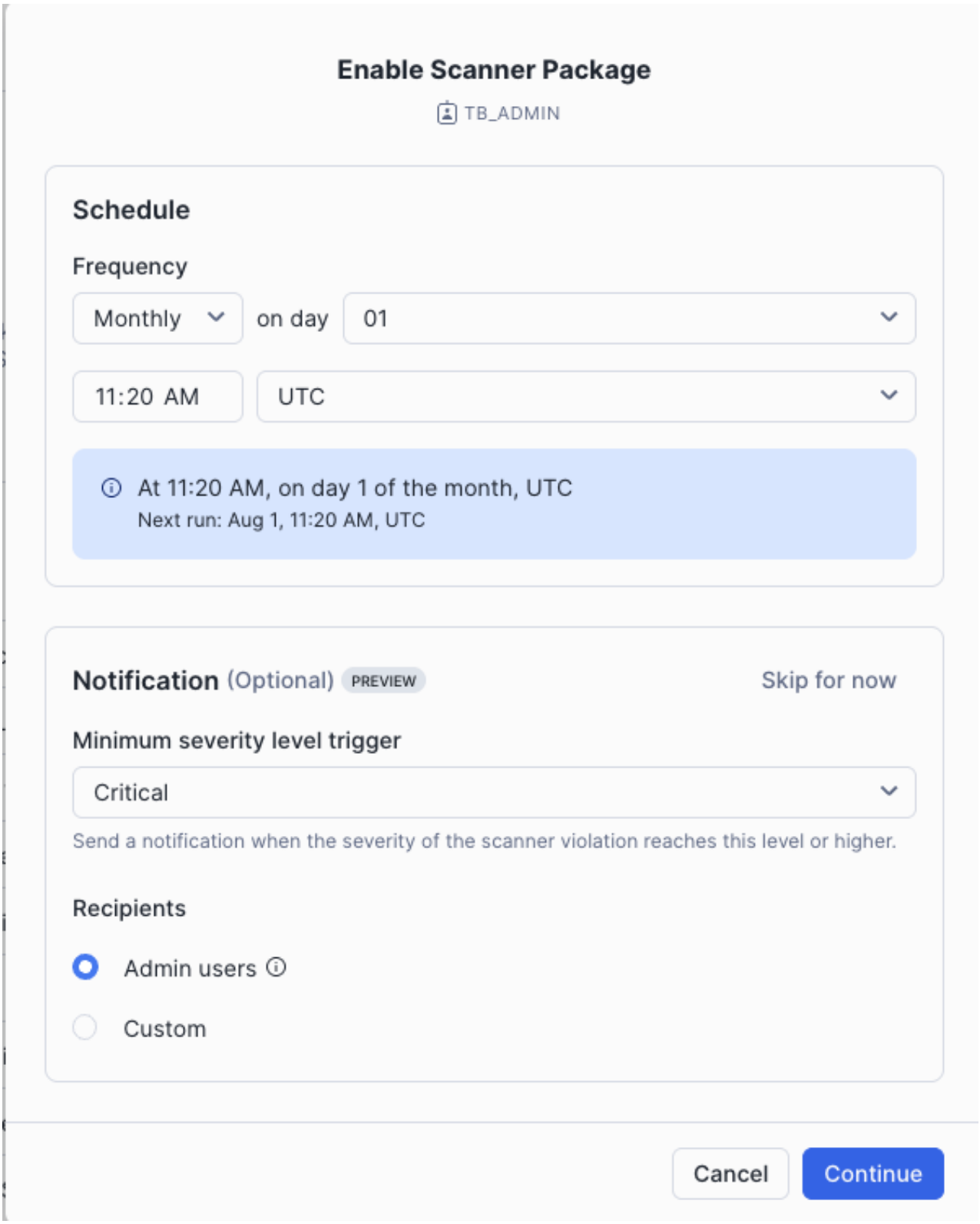
1. Trust Center で、**Scanner Packages** タブをクリックします。
2. **CIS Benchmarks** をクリックします。



3. **Enable Package** ボタンをクリックします。



4. モーダルで、**Frequency** を `Monthly` に設定し、**Continue** をクリックします。



5. **Threat Intelligence** スキャナーパッケージについても、このプロセスを繰り返します。

ステップ 3 - 調査結果の確認

スキャナーを実行する時間を与えた後、**Findings** タブに戻ります。

- 重大度別の違反をまとめたダッシュボードが表示されます。
- 以下のリストには、各違反、その重大度、およびそれを検出したスキャナーの詳細が表示されます。
- 違反をクリックすると、サマリーと推奨される修復手順を含む詳細ペインが開きます。

- 重大度、ステータス、またはスキャナーパッケージでリストをフィルタリングして、最も重要な問題に集中できます。