Appliquer des filtres aux requêtes SQL

Description du projet

Ma mission au sein de l'organisation est de renforcer la sécurité globale du système. Je m'occupe d'enquêter sur les failles potentielles et de mettre à jour les accès aux postes de travail pour limiter les risques.

Pour ce projet, j'ai utilisé **SQL** et des **filtres avancés** afin d'auditer précisément les données et d'automatiser les tâches de vérification liées à la sécurité.

Récupérer les tentatives de connexion infructueuses après les heures d'ouverture

Un incident de sécurité potentiel a été signalé après 18h00. Pour comprendre ce qui s'est passé, j'ai dû fouiller dans les logs de connexion pour isoler toutes les tentatives qui ont échoué en dehors des heures de bureau.

Ma démarche avec SQL :

J'ai construit une requête précise pour filtrer la table `log_in_attempts`. L'objectif était d'extraire uniquement les lignes répondant à deux critères critiques :

1. **L'horaire** : Les connexions tentées après **18:00**.
2. **L'échec** : Uniquement les tentatives infructueuses (`success = FALSE`).

```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_time > '18:00' AND success = FALSE;
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        2 | apatel   | 2022-05-10 | 20:27:27   | CAN     | 192.168.205.12  |       0 |
|       18 | pwashing | 2022-05-11 | 19:28:50   | US      | 192.168.66.142  |       0 |
|       20 | tshah    | 2022-05-12 | 18:56:36   | MEXICO  | 192.168.109.50  |       0 |
```

Ce que j'ai découvert : En utilisant la clause `WHERE` avec l'opérateur `AND`, j'ai pu identifier immédiatement trois tentatives suspectes (utilisateurs `apatel`, `pwashing`, et `tshah`). Cette étape est cruciale car elle permet de passer de milliers de lignes de logs à une poignée d'alertes exploitables pour une enquête plus approfondie.

Récupérer les tentatives de connexion à des dates spécifiques

Un événement suspect a été détecté le **09/05/2022**. Pour ne rien rater, j'ai décidé d'élargir l'enquête en vérifiant toutes les connexions survenues ce jour-là, mais aussi la veille.

```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_date = '2022-05-09' OR login_date = '2022-05-08';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.140 |       0 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.151.162 |       0 |
|        4 | dkot     | 2022-05-08 | 02:00:39   | USA     | 192.168.178.71  |       0 |
```

Ma démarche avec SQL :

L'idée était de ratisser large sur ces 48 heures critiques. J'ai utilisé la table `log_in_attempts` avec une condition flexible :

- J'ai appliqué une clause `WHERE` combinée à l'opérateur **`OR`**.
- Cela m'a permis de récupérer toutes les tentatives, qu'elles aient eu lieu le **09/05/2022** ou le **08/05/2022**.

Ce que j'ai découvert : Grâce à ce filtrage, j'ai pu isoler trois lignes précises (utilisateurs `jrafael` et `dkot`). Utiliser l'opérateur `OR` est ici indispensable : il permet de regrouper plusieurs critères de temps dans un seul rapport, ce qui est beaucoup plus rapide que de faire deux recherches séparées. C'est un gain de temps énorme quand on doit corréler des événements sur plusieurs jours.

Récupérer les tentatives de connexion effectuées en dehors du Mexique

En analysant les logs, j'ai remarqué que les activités provenant du Mexique ne semblaient pas correspondre au profil habituel de nos connexions. Pour isoler le problème, j'ai décidé de générer un rapport affichant toutes les tentatives de connexion effectuées **partout, sauf au Mexique**.

Ma démarche avec SQL : Pour ne pas avoir à lister manuellement tous les autres pays, j'ai utilisé une commande d'exclusion très pratique dans ma requête :

- J'ai utilisé l'opérateur **`NOT`** combiné à **`LIKE`**.
- Le filtre **`'MEX%'`** (avec le signe `%`) me permet de capturer toutes les variantes d'écriture du pays pour être sûr de ne rien laisser passer.

```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE NOT country LIKE 'MEX%';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.140 |       0 |
|        2 | apatel   | 2022-05-10 | 20:27:27   | CAN     | 192.168.205.12  |       0 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.151.162 |       0 |
```

Le résultat : Ma requête a instantanément isolé les connexions provenant du Canada (CAN) et des États-Unis (USA) qui méritaient une attention particulière. En cybersécurité, savoir exclure le "bruit" (les données normales d'une zone) est indispensable pour faire ressortir les anomalies réelles venant d'ailleurs.

Récupérer les employés en marketing

Mon équipe doit renouveler les ordinateurs des employés du service **Marketing** situés dans le **bâtiment Est**. Pour organiser l'intervention, j'ai dû extraire la liste précise des postes concernés.

Ma démarche avec SQL : J'ai interrogé la table `employees` en combinant deux critères de filtrage pour être ultra-précis :

1. **Le département** : J'ai filtré sur `'Marketing'`.
2. **La localisation** : Comme les numéros de bureau varient, j'ai utilisé **`LIKE 'East%'`**. Le symbole `%` m'a permis de capturer tous les bureaux du bâtiment Est (East-170, East-195, etc.) en une seule commande.

```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Marketing' AND office LIKE 'East%';
+-------------+-------------+----------+------------+----------+
| employee_id | device_id   | username | department | office   |
+-------------+-------------+----------+------------+----------+
|        1000 | a320b137c219 | elarson  | Marketing  | East-170 |
|        1052 | a192b174c940 | jdarosa  | Marketing  | East-195 |
|        1075 | x573y883z772 | fbautist | Marketing  | East-267 |
```

Le résultat : La requête a listé les trois employés concernés (`elarson`, `jdarosa`, `fbautist`) avec leurs identifiants d'appareils respectifs. C'est l'exemple type de la manière dont SQL permet de transformer une base de données massive en une liste de tâches concrète pour l'équipe technique.

Récupérer les employés des services financiers ou commerciaux

Pour terminer la mise à jour de sécurité, j'ai dû extraire les informations des employés travaillant dans deux services spécifiques : la Finance et les Ventes. L'objectif était d'obtenir une liste consolidée de leurs appareils.

Ma démarche avec SQL : Comme je devais cibler deux départements différents dans une seule recherche, j'ai utilisé une logique inclusive :

- J'ai interrogé la table `employees`.
- J'ai utilisé la clause `WHERE` avec l'opérateur **`OR`** : `department = 'Finance' OR department = 'Sales'`.

```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Finance' OR department = 'Sales';
+-------------+-------------+----------+------------+------------+
| employee_id | device_id   | username | department | office     |
+-------------+-------------+----------+------------+------------+
|        1003 | d394e816f943 | sgilmore | Finance    | South-153  |
|        1007 | h174i497j413 | wjaffrey | Finance    | North-406  |
|        1008 | i858j583k571 | abernard | Finance    | South-170  |
```

Le résultat : La requête a parfaitement fonctionné en affichant les employés des deux services (comme `sgilmore` ou `wjaffrey`). Utiliser `OR` à la place de `AND` est ici crucial : avec `AND`, la base chercherait quelqu'un travaillant dans les deux services en même temps, ce qui ne renverrait aucun résultat. C'est la méthode la plus propre pour regrouper plusieurs catégories dans un même rapport d'audit.

Récupérez tous les employés qui ne travaillent pas dans le service informatique.

Pour la toute dernière mise à jour de sécurité, mon équipe avait besoin des informations sur tous les employés, à l'exception de ceux du service informatique. J'ai donc dû générer une liste excluant spécifiquement ce département.

Ma démarche avec SQL : Au lieu de lister tous les autres services un par un, j'ai utilisé une méthode d'exclusion directe dans ma requête sur la table `employees` :

- J'ai appliqué la clause `WHERE` avec l'opérateur **`NOT`**.
- La condition était simple : `NOT department = 'Information Technology'`.

```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE NOT department = 'Information Technology';
+-------------+--------------+----------+-----------------+-------------+
| employee_id | device_id    | username | department      | office      |
+-------------+--------------+----------+-----------------+-------------+
|        1000 | a320b137c219 | elarson  | Marketing       | East-170    |
|        1001 | b239c825d303 | bmoreno  | Marketing       | Central-276 |
|        1002 | c116d593e558 | tshah    | Human Resources | North-434   |
```

Le résultat : La requête a instantanément regroupé les employés du Marketing, des RH, etc. (comme `elarson`, `bmoreno`, ou `tshah`). C'est l'outil parfait pour isoler rapidement un groupe spécifique et s'assurer que les politiques de sécurité sont appliquées partout ailleurs dans l'entreprise.

Résumé

Ce projet m'a permis de démontrer comment SQL devient un allié redoutable pour un analyste en cybersécurité. En jonglant avec les tables `log_in_attempts` et `employees`, j'ai réussi à :

- **Débusquer des anomalies** (connexions suspectes hors horaires ou géographies).
- **Cibler des interventions** (mises à jour de machines par bâtiment ou service).
- **Maîtriser les opérateurs logiques** (`AND`, `OR`, `NOT`) et les **caractères génériques** (`LIKE %`) pour transformer des milliers de données brutes en informations exploitables.