Apply filters to SQL queries

Project description

In my role as a security professional at a large organization, my primary responsibility is to enhance the security of our system. I diligently work towards ensuring its overall safety by proactively investigating potential security issues. Additionally, I actively monitor and update employee computers as necessary to maintain a secure environment. The subsequent steps exemplify how I effectively employ SQL filters to execute various security-related tasks, thereby bolstering the organization's security infrastructure.

Retrieve after hours failed login attempts

After business hours, there was a potential security incident that requires thorough investigation. Specifically, it is crucial to examine all failed login attempts that occurred outside regular working hours (after 18:00).

To accomplish this task, I employed SQL queries and filters to retrieve the necessary information. The provided code snippet showcases the query I formulated to filter for failed login attempts after 18:00.

```
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 |
```

The first part of the screenshot illustrates my query, while the second part displays a portion of the resulting output. The query begins by selecting all data from the log_in_attempts table. Subsequently, a WHERE clause utilizing the AND operator is utilized to filter the results. The first condition, login_time > '18:00', ensures that only login attempts after 18:00 are included. The second condition, success = FALSE, narrows down the results to only failed login attempts.

This approach allows for a focused analysis of potential security breaches that occurred during after-hours, enabling us to identify and address any vulnerabilities promptly.

Retrieve login attempts on specific dates

An incident of suspicious nature took place on 2022-05-09, prompting the need for a comprehensive investigation. As part of the investigation process, it is essential to examine all login activities that occurred on 2022-05-09 and the preceding day.

To accomplish this task, I formulated a SQL query with appropriate filters to retrieve login attempts that transpired on specific dates. The provided code snippet represents the query I constructed, while the accompanying screenshot displays a portion of the resulting output. The query retrieves all login attempts from the log_in_attempts table, utilizing a WHERE clause with an OR operator to filter the results based on the desired dates.

```
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 |
```

The first condition, login_date = '2022-05-09', ensures that only login attempts on 2022-05-09 are included. Similarly, the second condition, login_date = '2022-05-08', narrows down the results to login attempts on 2022-05-08.

This approach allows us to focus our investigation on login activities surrounding the suspicious event, aiding in the identification and analysis of any potential security breaches or anomalies that occurred on the specified dates.

Retrieve login attempts outside of Mexico

Upon thoroughly investigating the organization's login attempt data, I have identified a potential issue concerning login attempts originating outside of Mexico. It is crucial to thoroughly investigate these specific login attempts to ensure the system's security.

To address this concern, I crafted a SQL query utilizing appropriate filters to retrieve login attempts that occurred outside of Mexico. The provided code snippet showcases the query I

developed, while the accompanying screenshot presents a portion of the resulting output. The query effectively returns all login attempts recorded in countries other than Mexico.

```
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 |
```

In this query, I began by selecting all data from the log_in_attempts table. Subsequently, I employed a WHERE clause with the NOT operator to filter for countries other than Mexico. To match the country names, I utilized the LIKE operator with the pattern 'MEX%' since the dataset represents Mexico as either 'MEX' or 'MEXICO'. The percentage sign (%) employed with LIKE matches any number of unspecified characters, enabling us to capture various country name formats.

By implementing this approach, we can focus our investigation on login attempts originating outside of Mexico, providing us with valuable insights into potential security risks or unauthorized access attempts from foreign locations.

Retrieve employees in Marketing

In order to update the computers for specific employees in the Marketing department, my team requires accurate information regarding which employee machines should be targeted for the updates.

To accomplish this task, I devised a SQL query with appropriate filters to identify the employee machines belonging to the Marketing department within the East building. The provided code snippet illustrates the query I developed, and the corresponding screenshot showcases a portion of the resulting output. The query effectively retrieves all employees working in the Marketing department within the East building.

```
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 |
```

The query initiates by selecting all data from the employees table. Subsequently, a WHERE clause with the AND operator is utilized to filter the results based on two conditions. The first condition, department = 'Marketing', ensures that only employees from the Marketing department are included. The second condition, office LIKE 'East%', employs the LIKE operator with the pattern 'East%' to match the specific office number in the office column, representing the East building.

By implementing this query, we can obtain the necessary information on the employees' machines in the Marketing department within the East building. This will enable my team to proceed with the targeted computer updates, ensuring the appropriate resources are allocated to enhance the security and performance of their systems.

Retrieve employees in Finance or Sales

To ensure comprehensive updates, it is essential to address the machines of employees in both the Finance and Sales departments. As the required security update differs from the previous one, it is imperative to gather relevant information solely on employees from these two departments.

To accomplish this, I formulated a SQL query incorporating appropriate filters to identify the employee machines belonging to the Finance or Sales departments. The provided code snippet demonstrates the query I constructed, and the corresponding screenshot displays a portion of the resulting output. The query successfully retrieves all employees belonging to the Finance and Sales departments.

```
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   |
```

The query commences by selecting all data from the employees table. Subsequently, a WHERE clause with the OR operator is employed to filter the results based on two conditions. The first condition, department = 'Finance', ensures that only employees from the Finance department are included. Similarly, the second condition, department = 'Sales', filters for employees from the Sales department.

By implementing this query, we can obtain the necessary information on employee machines specifically within the Finance and Sales departments. This allows us to prioritize the updates required for these departments, ensuring that their systems remain secure and up to date.

Retrieve all employees not in IT

To proceed with an additional security update, it is necessary for my team to address the machines of employees who do not belong to the Information Technology (IT) department. To accomplish this, we must first obtain the relevant information regarding these employees.

The provided code snippet displays the query I devised, while the accompanying screenshot showcases a portion of the resulting output. This query effectively retrieves all employees who are not part of the Information Technology department.

```
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   |
```

The query starts by selecting all data from the employees table. It then employs a WHERE clause utilizing the NOT operator to filter for employees who do not belong to the Information Technology department. The condition department <> 'Information Technology' ensures that only employees outside of the IT department are included.

By implementing this query, we can gather the necessary information on employees who require the additional security update but are not associated with the IT department. This enables us to focus our efforts on these specific employees, ensuring the necessary security measures are implemented to protect their machines and the overall system integrity.

Summary

To extract specific information on login attempts and employee machines, I adeptly utilized filters in my SQL queries. The queries were executed on two distinct tables, namely log_in_attempts and employees. By strategically employing operators such as AND, OR, and NOT, I successfully filtered the data to retrieve the precise information required for each task.

Additionally, I leveraged the LIKE operator and the versatile percentage sign (%) wildcard to filter for patterns within the data. This enabled me to refine the results and extract relevant information based on specific matching criteria.

Through the judicious use of these filtering techniques and operators, I efficiently retrieved the necessary data and obtained actionable insights. This allowed me to investigate potential security issues, identify login attempts outside of specified regions, analyze login attempts on specific dates, and gather information on employees from specific departments.

In summary, my proficient use of filters and SQL query techniques facilitated the extraction of precise and relevant information from the log_in_attempts and employees tables, enabling effective investigation of security issues and informed decision-making to enhance system security.