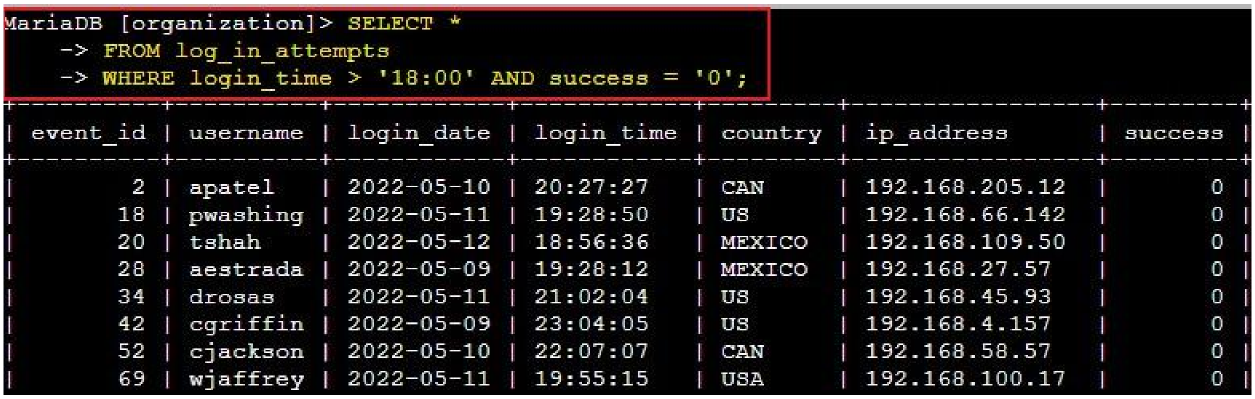
Apply filters to SQL queries

Project description

As a security analyst in a large organization, a crucial aspect of the job is to investigate potential security issues to maintain a secure system. Recently they noticed suspicious login attempts and concerns related to employee machines. To address these issues, they will utilize SQL queries to extract relevant records from the organization’s **`employees`** and **`log_in_attempts`** tables. By running these SQL filters, they can gather essential data from different sets of information and conduct a thorough investigation to identify and mitigate security risks, ensuring the overall security and integrity of the organization’s systems.

Retrieve after hours failed login attempts

- Failed login attempts after 18:00



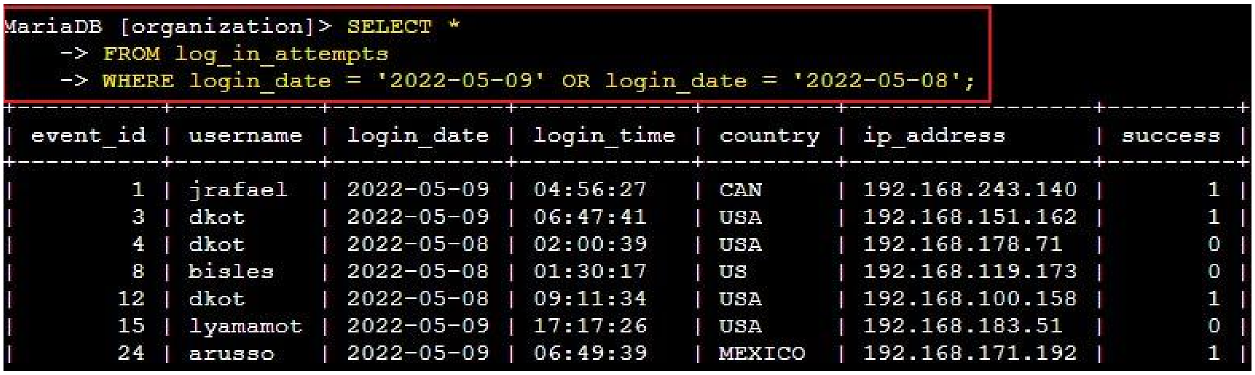
```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_time > '18:00' AND success = '0';
+----------+----------+------------+------------+---------+----------------+---------+
| event_id | username | login_date | login_time | country | ip_address     | success |
+----------+----------+------------+------------+---------+----------------+---------+
|        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 |
|       28 | aestrada | 2022-05-09 | 19:28:12   | MEXICO  | 192.168.27.57  |       0 |
|       34 | drosas   | 2022-05-11 | 21:02:04   | US      | 192.168.45.93  |       0 |
|       42 | cgriffin | 2022-05-09 | 23:04:05   | US      | 192.168.4.157  |       0 |
|       52 | cjackson | 2022-05-10 | 22:07:07   | CAN     | 192.168.58.57  |       0 |
|       69 | wjaffrey | 2022-05-11 | 19:55:15   | USA     | 192.168.100.17 |       0 |
```

- As a security analyst, the focus is on narrowing down potential security threats by filtering SQL queries based on two key conditions. Firstly, I will focus on incidents that occurred after business hours, specifically from 18:00 (6:00pm) onward. Secondly, I will include only those queries that involve failed login attempts. By applying these filters, I can focus on investigating suspicious activities that might pose a security risk to the system.
    - **`SELECT*`**: indicates it will return all columns in the table.
    - **`FROM log_in_attempts`**: this indicates the table the security analyst wants to query.
    - **`WHERE login_time > ‘18:00’`**:this indicates the condition of the filter and this case the first condition is the login time being greater than 18:00 (6:00pm).
    - **`AND:`** is used to filter two conditions. It specifies that both conditions must be met simultaneously.

    - `success = ‘0’;`: this will be the second condition of the WHERE filter. Success equal to zero means the number of failed login attempts.
- After executing the command, the list will display all failed login attempts after 6:00pm on different dates.

Retrieve login attempts on specific dates

- Login attempts on specific dates

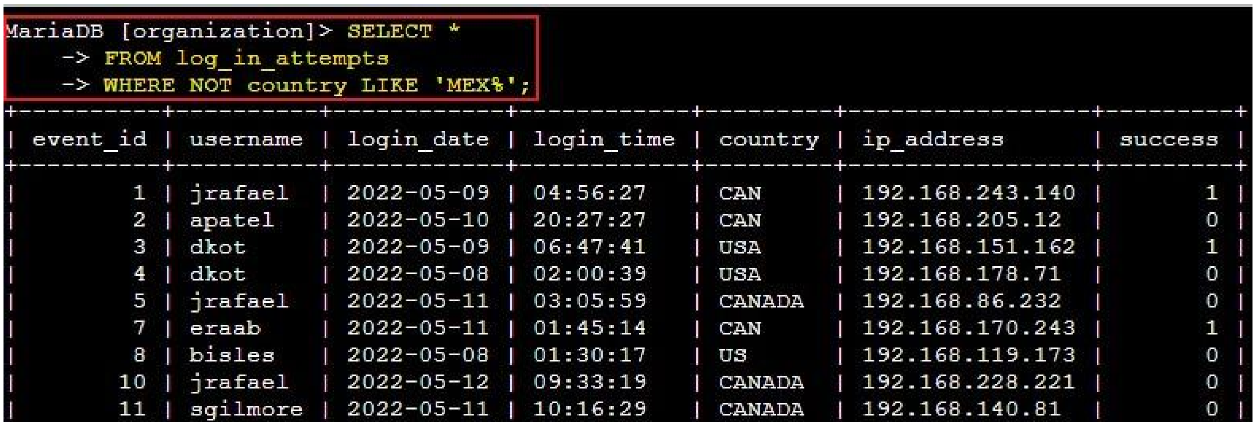


```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_date = '2022-05-09' OR login_date = '2022-05-08';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.140 |       1 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.151.162 |       1 |
|        4 | dkot     | 2022-05-08 | 02:00:39   | USA     | 192.168.178.71  |       0 |
|        8 | bisles   | 2022-05-08 | 01:30:17   | US      | 192.168.119.173 |       0 |
|       12 | dkot     | 2022-05-08 | 09:11:34   | USA     | 192.168.100.158 |       1 |
|       15 | lyamamot | 2022-05-09 | 17:17:26   | USA     | 192.168.183.51  |       0 |
|       24 | arusso   | 2022-05-09 | 06:49:39   | MEXICO  | 192.168.171.192 |       1 |
```

- As the security analyst, filtering the SQL query to review all login attempts on 2022-05-09 and the day before (2022-05-08). By comparing these two days of login attempts, I can detect any anomalies or unusual patterns that might indicate potential security threats or suspicious activities related to the event on 2022-05-09. This focused analysis will help investigate the incident thoroughly and identify any potential risks to the system.
    - `SELECT*`: indicates it will return all columns in the table.
    - `FROM log_in_attempts`: this indicates the table the security analyst wants to query.
    - `WHERE login_date = ‘2022-05-09’`: this indicates the condition of the filter and in this case the first condition is for the amount of login attempts on `login_date` that is equal to May 9th, 2022 (`‘2022-05-09’`).
    - `OR`: this operator also connects two conditions, but `OR` specifies that either condition can be met.
    - `login_date = ‘2022-05-08’;`: this indicates the condition of the filter and in this case the second condition is for the amount of login attempts on `login_date` that is equal to May 8th, 2022 (`‘2022-05-08’`).
- After executing the command, the query will display all login attempts whether successful or failed on May 8th 2022 and May 9th 2022.

Retrieve login attempts outside of Mexico

- Login attempts outside of Mexico

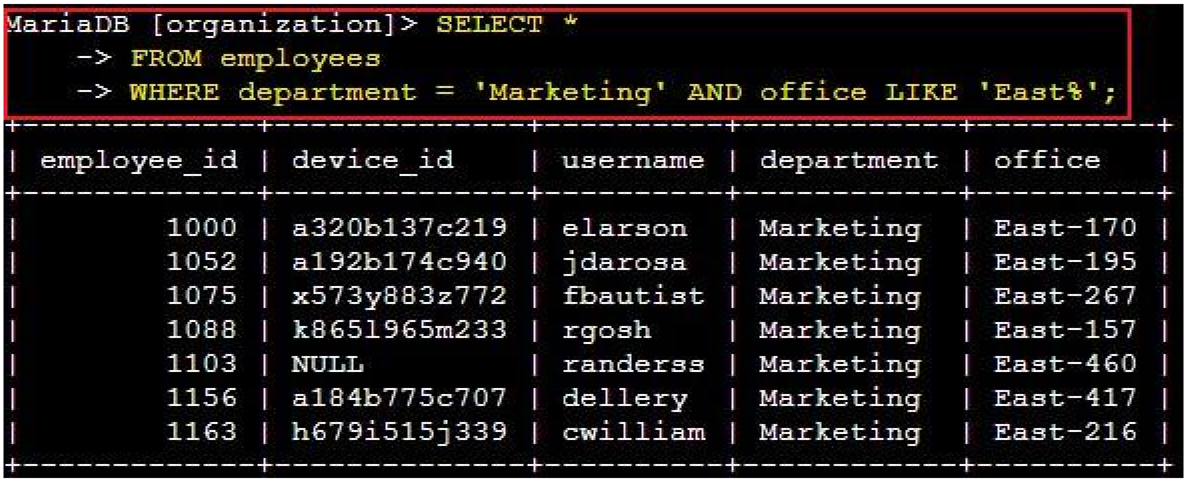


```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE NOT country LIKE 'MEX%';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.140 |       1 |
|        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 |       1 |
|        4 | dkot     | 2022-05-08 | 02:00:39   | USA     | 192.168.178.71  |       0 |
|        5 | jrafael  | 2022-05-11 | 03:05:59   | CANADA  | 192.168.86.232  |       0 |
|        7 | eraab    | 2022-05-11 | 01:45:14   | CAN     | 192.168.170.243 |       1 |
|        8 | bisles   | 2022-05-08 | 01:30:17   | US      | 192.168.119.173 |       0 |
|       10 | jrafael  | 2022-05-12 | 09:33:19   | CANADA  | 192.168.228.221 |       0 |
|       11 | sgilmore | 2022-05-11 | 10:16:29   | CANADA  | 192.168.140.81  |       0 |
```

- To investigate the suspicious login attempts, the security team will create an SQL query to filter out login attempts that did not originate in Mexico. By analyzing the location data associated with each login attempt, the team can identify all login activities that occurred outside of Mexico. This narrowed-down query will provide crucial information to focus on potentially malicious activities and aid in their investigation to assess and respond to the security threat effectively.
    - `SELECT*`: indicates it will return all columns in the table.
    - `FROM log_in_attempts`: this indicates the table the security analyst wants to query regarding login attempts.
    - `WHERE NOT country LIKE ‘MEX%’;`:this indicates the condition of the filter and in this case the condition is to not include a specified country. The `LIKE` filter mixed with the `%` wildcard will include the title of the country whether it is spelled `‘MEXICO’` or `‘MEX’`.
- After executing the command, the query will display all countries that are not named `MEXICO` or `MEX`.

Retrieve employees in Marketing

- Employees in Marketing

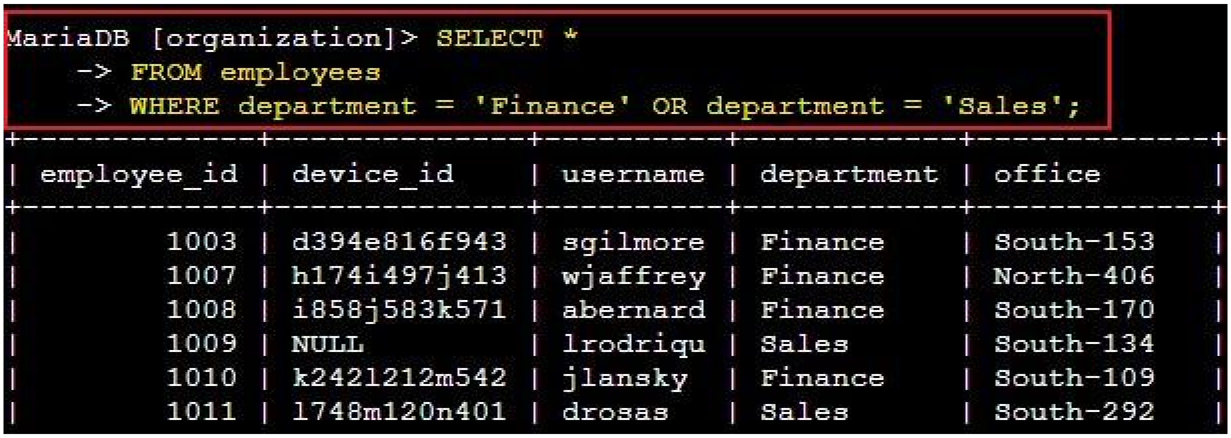


```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Marketing' AND office LIKE 'East%';
+-------------+---------------+----------+------------+-----------+
| employee_id | device_id     | username | department | office    |
+-------------+---------------+----------+------------+-----------+
|        1000 | a320b137c219  | elarson  | Marketing  | East-170  |
|        1052 | a192b174c940  | jdarosa  | Marketing  | East-195  |
|        1075 | x573y883z772  | fbautist | Marketing  | East-267  |
|        1088 | k8651965m233  | rgosh    | Marketing  | East-157  |
|        1103 | NULL          | randerss | Marketing  | East-460  |
|        1156 | a184b775c707  | dellery  | Marketing  | East-417  |
|        1163 | h679i515j339  | cwilliam | Marketing  | East-216  |
+-------------+---------------+----------+------------+-----------+
```

- To gather information on employee machines in the Marketing department of all offices in the East building, the security team will create a SQL query. This query will focus on filtering employee records based on their department (Marketing) and location (East building). By executing this SQL query on the employee database, the security analyst can obtain a list of employees whose machines need security updates, making it easier to perform targeted security measures and ensure the protection of sensitive data and systems within the Marketing department in the East building.
    - `SELECT*`: indicates it will return all columns in the table.
    - `FROM employees`: this indicates the table the security analyst wants to query.
    - `WHERE department = ‘Marketing’`: this indicates the condition of the filter and in this case the first condition is the department of `‘Marketing’` is included in the headcount of employees for this query.
    - `AND`: is used to filter two conditions. It specifies that both conditions must be met simultaneously.
    - `Office LIKE ‘East%’;`: this indicates the condition of the filter and in this case the second condition is the office includes only the East building with all floors.
- After executing the command, the query will return results showing all Marketing employees in the East building on any floor.

Retrieve employees in Finance or Sales

- Employees in Finance or Sales

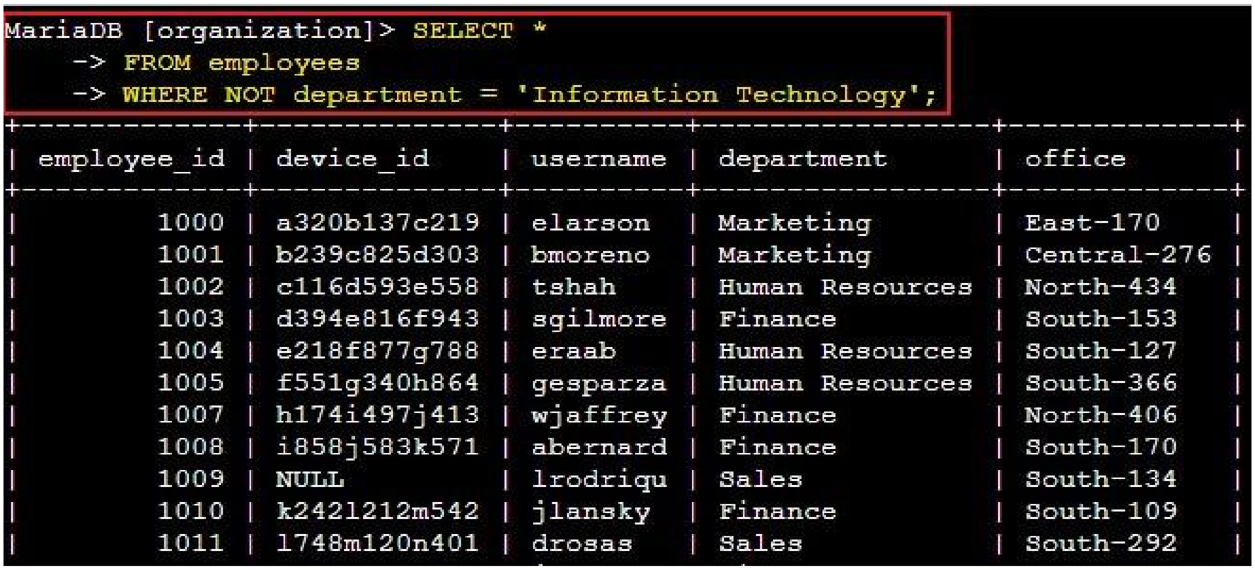


```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Finance' OR department = 'Sales';
+-------------+---------------+----------+------------+-------------+
| employee_id | device_id     | username | department | office      |
+-------------+---------------+----------+------------+-------------+
|        1003 | d394e816f943  | sgilmore | Finance    | South-153   |
|        1007 | h174i497j413  | wjaffrey | Finance    | North-406   |
|        1008 | i858j583k571  | abernard | Finance    | South-170   |
|        1009 | NULL          | lrodriqu | Sales      | South-134   |
|        1010 | k2421212m542  | jlansky  | Finance    | South-109   |
|        1011 | 1748m120n401  | drosas   | Sales      | South-292   |
```

- To identify all employees in the Sales and Finance departments, the security team will create a SQL query. This query will filter employee records based on their respective departments (Sales and Finance). By executing this SQL query on the employee database, the team can obtain a comprehensive list of employees belonging to the Sales and Finance departments, allowing them to perform specific security updates on their machines to enhance the security measures tailored to each department's needs. This focused approach ensures that the security updates are accurately applied to the relevant systems, minimizing potential risks and vulnerabilities.
    - **`SELECT*`**: indicates it will return all columns in the table.
    - **`FROM employees`**: this indicates the table the security analyst wants to query.
    - **`WHERE department = ‘Finance’`**: this indicates the condition of the filter and in this case the first condition is the department of **`‘Finance’`** is included in the headcount of employees for this query.
    - **`OR`**: this operator also connects two conditions, but **`OR `** specifies that either condition can be met.
    - **`department = ‘Sales’;`**: this indicates the condition of the filter and in this case the second condition is the department of **`‘Sales’`** is included in the headcount of employees for this query.
- After executing the command, the query will return results showing all Finance and Sales employees in the organization no matter the building.

Retrieve all employees not in IT

- Employees not in IT



```
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   |
|        1003 | d394e816f943  | sgilmore | Finance         | South-153   |
|        1004 | e218f877g788  | eraab    | Human Resources | South-127   |
|        1005 | f551g340h864  | gesparza | Human Resources | South-366   |
|        1007 | h174i497j413  | wjaffrey | Finance         | North-406   |
|        1008 | i858j583k571  | abernard | Finance         | South-170   |
|        1009 | NULL          | lrodriqu | Sales           | South-134   |
|        1010 | k2421212m542  | jlansky  | Finance         | South-109   |
|        1011 | 1748m120n401  | drosas   | Sales           | South-292   |
```

- To identify all employees not in the Information Technology (IT) department, the security team will create a SQL query. This query will filter employee records based on their department, excluding those who belong to the IT department. By executing this SQL query on the employee database, the team can obtain a comprehensive list of employees who still need the security update, ensuring that the update is applied to all relevant machines and minimizing any potential security gaps. This targeted approach helps avoid unnecessary updates for employees who have already received the update, streamlining the security update process.
    - `SELECT*`: indicates it will return all columns in the table.
    - `FROM employees`: this indicates the table the security analyst wants to query.
    - `WHERE NOT department = ‘Information Technology’;`: this indicates the condition of the filter and in this case the condition is to not include the department `‘Information Technology’` in this query.
- After executing the command, the query will return results showing all employees who are not in the Information Technology department.

Summary

The security team successfully utilized SQL queries to conduct a comprehensive investigation, narrowing down potential security incidents and identifying appropriate courses of action to mitigate risks effectively. They also implemented targeted security updates for different departments, ensuring that each area received the necessary protection and minimizing potential risks and vulnerabilities. Through their diligent use of SQL queries, the team demonstrated their ability to proactively address security concerns and maintain a secure environment for the organization.