Investigating and Mitigating Security Threats through SQL Queries

Project description

As a security professional in a large organization, my role involves monitoring and investigating security incidents to protect the company's systems. In this project, I conducted an in-depth SQL-based security investigation to identify potential security threats related to login attempts and employee machines. By analyzing data from the employees and log_in_attempts tables, I aimed to detect suspicious activities and mitigate risks.

Retrieve after hours failed login attempts

During routine security monitoring, I discovered irregularities in login attempts that could indicate unauthorized access attempts, brute-force attacks, or compromised employee credentials. To further investigate, I queried the organization's databases to retrieve all failed login attempts after business hours.

```
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      | 192.168.4.1
57   |       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 |
|       82 | abernard | 2022-05-12 | 23:38:46   | MEX     | 192.168.234
.49  |       0 |
```

The screenshot above illustrates how I queried the organization's database to retrieve records from the log_in_attempts table, focusing on login attempts made after business hours. In the first line of the query, I used SELECT * to extract all columns from the table, as indicated by the FROM statement in the second line. To filter the results, I utilized the > and AND operators, specifying login_time > 18:00 to capture login attempts occurring after business hours and success = 0 to include only failed attempts. The AND operator was used to combine these conditions. In my organization's database system, boolean values are represented with 1 for TRUE and 0 for FALSE.

Retrieve login attempts on specific dates

While analyzing the data, my team and I identified a suspicious trend on **May 9, 2022**. To further investigate this event, I queried the database to retrieve all login attempts from **May 9, 2022**, as well as the previous day, **May 8, 2022**.

```
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 |
|       25 | sbaelish | 2022-05-09 | 07:04:02   | US      | 192.168.33.
137  |       1 |
|       26 | apatel   | 2022-05-08 | 17:27:00   | CANADA  | 192.168.123
.105 |       1 |
```

As shown in the screenshot, I used the SELECT clause in the first line to retrieve all columns from the log_in_attempts table, as specified in the second line with the FROM statement. To filter the data, I applied the WHERE clause, ensuring that only login attempts from **May 8 and May 9, 2022** were returned.

Retrieve login attempts outside of Mexico

To deepen our investigation, we needed to refine our search by excluding login attempts originating from Mexico. This decision was made because initial findings showed a high volume of expected logins from Mexico, which could overshadow potential anomalies in other locations. By filtering out these logins, we aimed to focus on regions with unusual or unexpected activity that might be linked to the suspicious trend identified on **May 8 and May 9, 2022**.

```
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 |
```

As shown in the screenshot, I used the SELECT clause in the first line to retrieve all columns from the log_in_attempts table, as indicated in the second line with the FROM statement. To exclude Mexico, I applied the WHERE clause in the third line, utilizing the NOT operator along

with LIKE and the % wildcard to filter out records where the country was recorded as "MEX." This allowed us to better isolate and analyze login attempts from other regions while continuing our investigation.

Retrieve employees in Marketing

As our investigation progressed, we discovered that a **password attack** had been used to brute-force the credentials of employees in certain departments. Amongst are the **Marketing** department located in the **East building offices** (e.g., "East-170" or "East-320"). To address this security concern and assist with updating employee machines, we needed to retrieve information on all affected employees.

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

As shown in the screenshot, I used the SELECT clause in the first line to retrieve all columns from the employee table, as specified in the second line with the FROM statement. To filter the data for only employees in the **Marketing** department, I applied the WHERE clause in the third line of the query. Additionally, I used the AND operator along with the % wildcard to include all offices in the **East building**, ensuring that we captured all relevant employee records.

Retrieve employees in Finance or Sales

After successfully updating the **Marketing department's machines**, we conducted a **security audit** to identify any remaining vulnerabilities that could be exploited. During our **penetration testing**, we discovered that machines belonging to employees in the **Finance and Sales departments** were vulnerable. To mitigate this risk, our team needed to **perform a separate update** on the computers of all employees in these 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   |
|        1009 | NULL         | lrodriqu | Sales      | South-134   |
|        1010 | k242l212m542 | jlansky  | Finance    | South-109   |
|        1011 | l748m120n401 | drosas   | Sales      | South-292   |
|        1015 | p611q262r945 | jsoto    | Finance    | North-271   |
|        1017 | r550s824t230 | jclark   | Finance    | North-188   |
|        1018 | s310t540u653 | abellmas | Finance    | North-403   |
|        1022 | w237x430y567 | arusso   | Finance    | West-465    |
|        1024 | y976z753a267 | iuduike  | Sales      | South-215   |
|        1025 | z381a365b233 | jhill    | Sales      | North-115   |
|        1029 | d336e475f676 | ivelasco | Finance    | East-156    |
|        1035 | j236k3031245 | bisles   | Sales      | South-171   |
|        1039 | n253o917p623 | cjackson | Sales      | East-378    |
|        1041 | p929q222r778 | cgriffin | Sales      | North-208   |
|        1044 | s429t157u159 | tbarnes  | Finance    | West-415    |
|        1045 | t567u844v434 | pwashing | Finance    | East-115    |
|        1046 | u429v921w138 | daquino  | Finance    | West-280    |
|        1047 | v109w587x644 | cward    | Finance    | West-373    |
|        1048 | w167x592y375 | tmitchel | Finance    | South-288   |
```

As shown in the screenshot, I used the SELECT clause in the first line to retrieve all columns from the employees table, as indicated in the second line with the FROM statement. To filter the data and retrieve information on **all employees in the Finance and Sales departments**, I applied the WHERE clause in the third line of the query. Since no employees work in both departments simultaneously, I used the OR operator to ensure the query **returned results for employees from either department**.

This allowed us to efficiently locate all affected employees in the **Finance and Sales departments**, ensuring that their systems were updated to **strengthen security and prevent potential exploits**.

Retrieve all employees not in IT

Lastly, we needed to **update the machines of employees in the IT department**, as these systems have **privileged access** that, if compromised, could cause severe damage to the organization's networks. This update was crucial to ensure **up-to-date patches, baseline**

configurations, and properly configured firewalls to block unused ports, among other security measures.

After completing the IT department updates, our team needed to **gather information on all employees who were not part of the IT department** to verify that all other departments had also received necessary security measures.

```
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
 |
```

As shown in the screenshot above, I used the SELECT clause in the first line of the query to retrieve all columns from the employees table, as indicated in the second line with the FROM statement. To filter the data and retrieve **only employees who were not in the IT department**, I used the WHERE clause with the NOT operator.

This final query allowed us to **confirm that all other employees remained accounted for in our security updates**, ensuring that the organization's systems were well-protected against potential threats.

Summary

As a **security professional**, I was responsible for investigating **potential security threats** within the organization's network. This involved analyzing login attempts, identifying compromised employee machines, and ensuring that security updates were applied accordingly. To achieve this, I queried the organization's database, specifically the `employees` and `log_in_attempts` tables, to extract relevant data using SQL filters. This project demonstrated how **SQL queries** can be used to **proactively investigate security threats, detect anomalies, and strengthen an organization's cybersecurity posture**. By leveraging data-driven insights, our team successfully **identified compromised systems, mitigated risks, and reinforced security measures** across multiple departments.