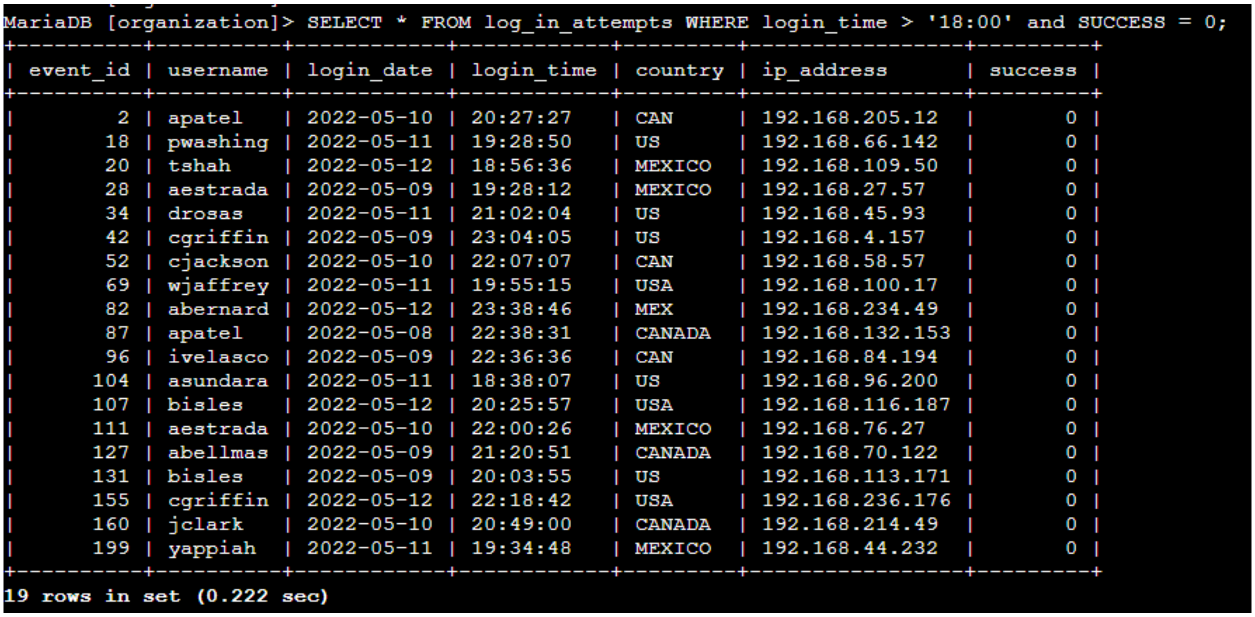
Apply filters to SQL queries

Project description

This project is to use the command line to practice applying filters in SQL. This is in order to refine queries. It must be noted that screenshots often contain only the first results of the query in order to keep the demonstration concise. The scenario is “You are a security professional at a large organization. Part of your job is to investigate security issues to help keep the system secure. You recently discovered some potential security issues that involve login attempts and employee machines.
Your task is to examine the organization’s data in their employees and log_in_attempts tables. You’ll need to use SQL filters to retrieve records from different datasets and investigate the potential security issues.”

Retrieve after hours failed login attempts

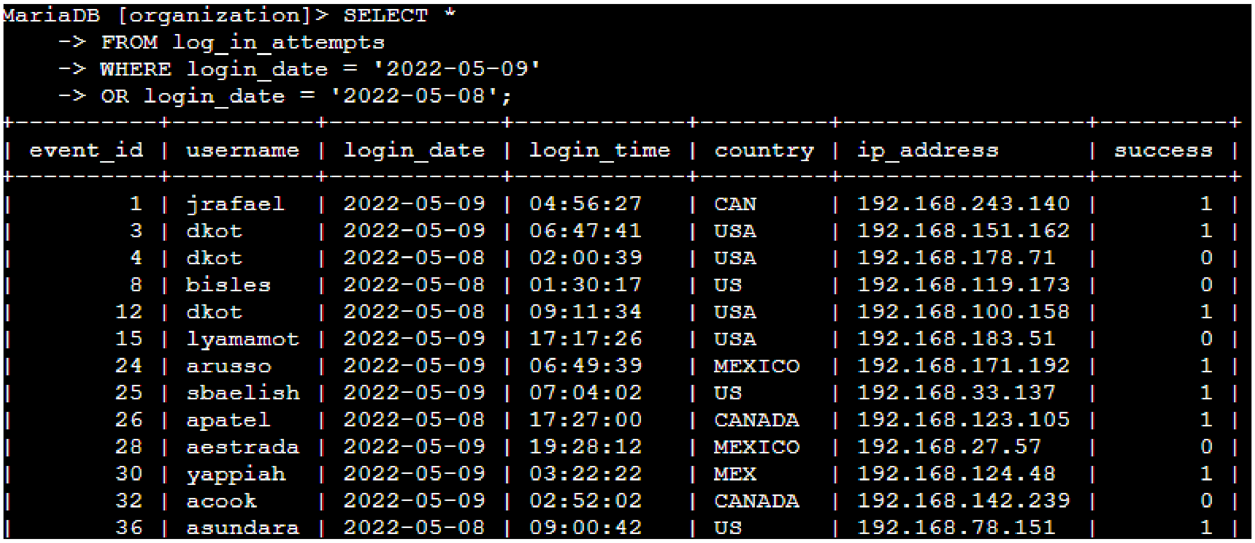
In order to explore the potential security issues, I needed to query the log_in_attempts table and review after hours login activity. By applying filters in SQL I created a query that identifies all failed login attempts that occurred after 18:00 [login_time > ‘18:00’ and SUCCESS = 0]. As can be seen, the time of the login attempt is found in the login_time column and we can see that the filter was applied successfully - all times are after 6pm. The success column contains a binary value of 0 when a login attempt failed or a 1 when attempts were successful. Here I filtered for all unsuccessful attempts. By applying the WHERE and AND filters I was able to narrow the results.



```
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 |
|       82 | abernard | 2022-05-12 | 23:38:46   | MEX     | 192.168.234.49  |       0 |
|       87 | apatel   | 2022-05-08 | 22:38:31   | CANADA  | 192.168.132.153 |       0 |
|       96 | ivelasco | 2022-05-09 | 22:36:36   | CAN     | 192.168.84.194  |       0 |
|      104 | asundara | 2022-05-11 | 18:38:07   | US      | 192.168.96.200  |       0 |
|      107 | bisles   | 2022-05-12 | 20:25:57   | USA     | 192.168.116.187 |       0 |
|      111 | aestrada | 2022-05-10 | 22:00:26   | MEXICO  | 192.168.76.27   |       0 |
|      127 | abellmas | 2022-05-09 | 21:20:51   | CANADA  | 192.168.70.122  |       0 |
|      131 | bisles   | 2022-05-09 | 20:03:55   | US      | 192.168.113.171 |       0 |
|      155 | cgriffin | 2022-05-12 | 22:18:42   | USA     | 192.168.236.176 |       0 |
|      160 | jclark   | 2022-05-10 | 20:49:00   | CANADA  | 192.168.214.49  |       0 |
|      199 | yappiah  | 2022-05-11 | 19:34:48   | MEXICO  | 192.168.44.232  |       0 |
+----------+----------+------------+------------+---------+-----------------+---------+
19 rows in set (0.222 sec)
```

Retrieve login attempts on specific dates

It became evident that a suspicious event occurred on 2022-05-09 and to investigate this event I needed to review all login attempts which occurred on this day and the day before.

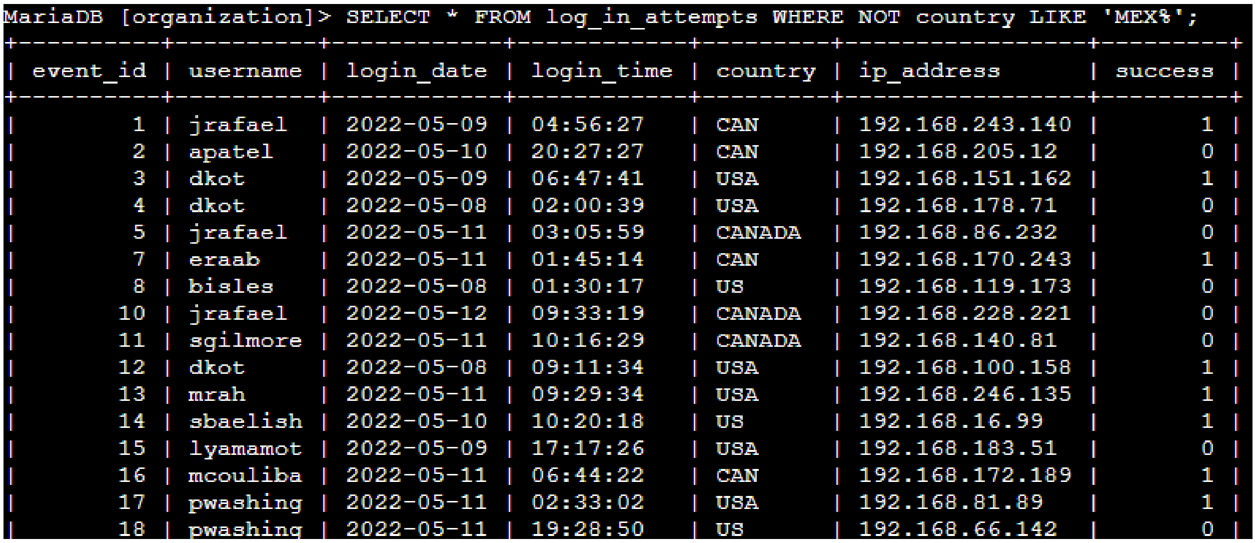


```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_date = '2022-05-09'
    -> OR login_date = '2022-05-08';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | 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 |
|       28 | aestrada | 2022-05-09 | 19:28:12   | MEXICO  | 192.168.27.57   |       0 |
|       30 | yappiah  | 2022-05-09 | 03:22:22   | MEX     | 192.168.124.48  |       1 |
|       32 | acook    | 2022-05-09 | 02:52:02   | CANADA  | 192.168.142.239 |       0 |
|       36 | asundara | 2022-05-08 | 09:00:42   | US      | 192.168.78.151  |       1 |
```

I’ve only included a screenshot of the first rows of data for demonstration, however using the filter WHERE to establish the first login date and OR to establish login attempts made on the previous date, it was established that 75 login attempts were made all day on those two days.

Retrieve login attempts outside of Mexico

It is established by the cybersecurity team that the potential activity did not originate in Mexico, so the goal of the next search is to keep only data that pertains to all countries outside of mexico.

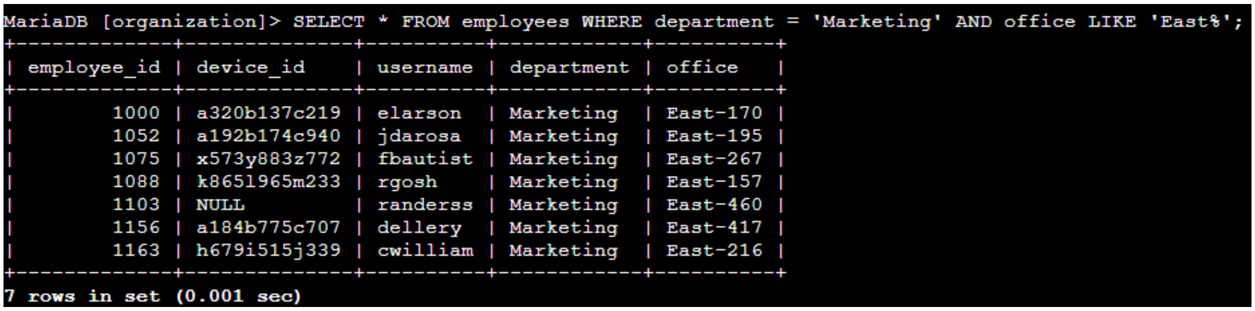


```
MariaDB [organization]> SELECT * FROM log_in_attempts WHERE NOT country LIKE 'MEX%';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        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 |
|       12 | dkot     | 2022-05-08 | 09:11:34   | USA     | 192.168.100.158 |       1 |
|       13 | mrah     | 2022-05-11 | 09:29:34   | USA     | 192.168.246.135 |       1 |
|       14 | sbaelish | 2022-05-10 | 10:20:18   | US      | 192.168.16.99   |       1 |
|       15 | lyamamot | 2022-05-09 | 17:17:26   | USA     | 192.168.183.51  |       0 |
|       16 | mcouliba | 2022-05-11 | 06:44:22   | CAN     | 192.168.172.189 |       1 |
|       17 | pwashing | 2022-05-11 | 02:33:02   | USA     | 192.168.81.89   |       1 |
|       18 | pwashing | 2022-05-11 | 19:28:50   | US      | 192.168.66.142  |       0 |
```

By using the WHERE NOT command I was able to narrow the records down to only those that are based in other countries. By using LIKE filter and MEX followed by the wildcard [%] I made sure this accounted for all records that represented Mexico because there were inconsistent representations of Mexico in the table (MEXICO/MEX).

Retrieve employees in Marketing

In the scenario I was asked to perform security updates on specific employee machines in the Marketing department who were located in the East offices.

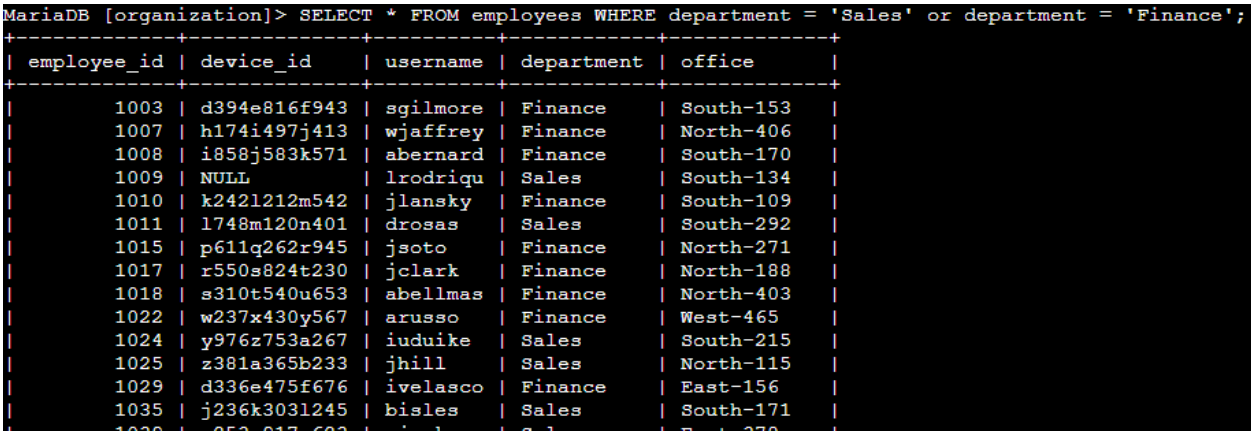


```
MariaDB [organization]> SELECT * FROM employees WHERE department = 'Marketing' AND office LIKE 'East%';
+-------------+---------------+----------+------------+----------+
| employee_id | device_id     | username | department | office   |
+-------------+---------------+----------+------------+----------+
|        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 |
+-------------+---------------+----------+------------+----------+
7 rows in set (0.001 sec)
```

By applying the WHERE to the department column and LIKE (East%) to the office column, I was able to make sure that only the entries for those employees in marketing working in the various East offices were included in the results.

Retrieve employees in Finance or Sales

I was requested to perform a different security update on machines for employees in the Sales and Finance departments. Here I queried the employees table to get only those employees working in either Sales or Finance, as listed in the department column.

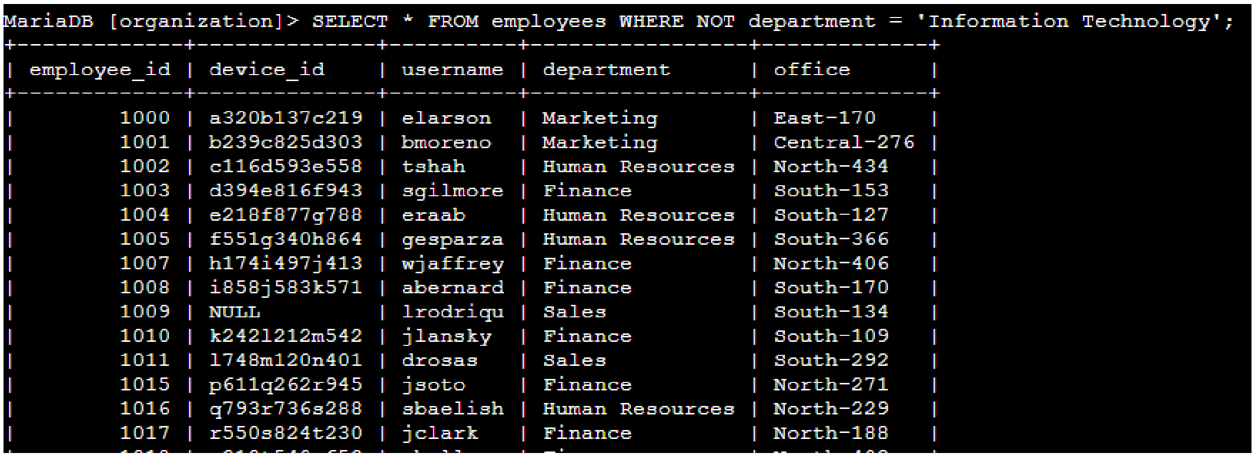


```
MariaDB [organization]> SELECT * FROM employees WHERE department = 'Sales' or department = 'Finance';
+-------------+---------------+----------+------------+-------------+
| employee_id | device_id     | username | department | office      |
+-------------+---------------+----------+------------+-------------+
|        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   |
|        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   |
```

By using the OR filter, I was able to get both departments, rather than one or the other obtained in AND queries (OR is used to return multiple queries).

Retrieve all employees not in IT

It has been requested that I make one more update to employee machines, however the employees who are in the Information Technology department already had this update, but employees in all other departments need 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    |
|        1015 | p611q262r945  | jsoto    | Finance         | North-271    |
|        1016 | q793r736s288  | sbaelish | Human Resources | North-229    |
|        1017 | r550s824t230  | jclark   | Finance         | North-188    |
```

By applying the WHERE NOT filter to the department column in the employees table I was able to narrow the results down to those who need the updates, ie. those who aren’t in the Information Technology department.

Summary

By using WHERE, WHERENOT, AND, OR as well as wildcard characters, data can be efficiently filtered in SQL to get the relevant results. In the scenario I was able to narrow down dates and times to investigate potential security vulnerabilities. I was able to exclude locations to include only those that were involved in potential security incidents. I was also able to identify specific departments and locations requiring updates. In security it’s important that software stays up-to-date so that security vulnerabilities from old editions can’t be exploited.