Google Cybersecurity Certificate Portfolio

Table of contents:

Controls and compliance checklist

Project description

Use the provided resources to conduct an internal security audit on a fictional company.

Controls assessment checklist

Yes	No	Control
	●	Least Privilege
●	●	Disaster recovery plans
●	●	Password policies
●	●	Separation of duties
●	●	Firewall
●	●	Intrusion detection system (IDS)
●	●	Backups
●	●	Antivirus software
●	●	Manual monitoring, maintenance, and intervention for legacy systems
	●	Encryption
●	●	Password management system
●	●	Locks (offices, storefront, warehouse)
●	●	Closed-circuit television (CCTV) surveillance
●	●	Fire detection/prevention (fire alarm, sprinkler system, etc.)

Compliance checklist

Payment Card Industry Data Security Standard (PCI DSS)

Yes	No	Best practice
	●	Only authorized users have access to customers' credit card information.
●	●	Credit card information is stored, accepted, processed, and transmitted internally, in a secure environment.
●	●	Implement data encryption procedures to better secure credit card transaction touchpoints and data.
●	●	Adopt secure password management policies.

General Data Protection Regulation (GDPR)

Yes	No	Best practice
	●	E.U. customers' data is kept private/secured.
●	●	There is a plan in place to notify E.U. customers within 72 hours if their data is compromised/there is a breach.
●	●	Ensure data is properly classified and inventoried.
●	●	Enforce privacy policies, procedures, and processes to properly document and maintain data.

System and Organizations Controls (SOC type 1, SOC type 2)

Yes	No	Best practice
	●	User access policies are established.
●	●	Sensitive data (PII/SPII) is confidential/private.
●	●	Data integrity ensures the data is consistent, complete, accurate, and has been validated.
●	●	Data is available to individuals authorized to access it.

Recommendations:

To minimize risk, implementing the following outstanding controls is advised: principle of least privilege, disaster recovery plan, separation of duties, install intrusion detection systems, data backups, customer data encryption, password management system and data classification and appropriate inventory.

Botium Toys should also improve their existing password policy to require more secure passwords and the monitoring and maintenance of legacy systems should be documented with appropriate schedules and interventions.

Cybersecurity Incident Report: Network Traffic Analysis

Project description

Analyse DNS and ICMP traffic in transit using data from a network protocol analyser tool (tcpdump).

Part 1: Provide a summary of the problem found in the DNS and ICMP traffic log.
The UDP protocol reveals that: port 53 is unreachable. This is based on the results of the network analysis, which show that the ICMP echo reply returned the error message: “udp port 53 unreachable”. The port noted in the error message is used for DNS protocol traffic. This suggests that there is an error with the DNS server.

Part 2: Explain your analysis of the data and provide at least one cause of the incident.
At 13:24:32 the team was alerted to the incident by the client, who informed us that their customers are not able to reach their website and are receiving the error message: “destination port unreachable”. The IT team used a network protocol analyzer, tcpdump, to analyze the network packet data. The results of the analysis were then reported to the department supervisor. The tcpdump analysis showed that port 53 on server with IP address: 203.0.113.2 is unreachable. The IT team will now investigate whether the DNS server is down or port 53 is blocked by the firewall. The DNS server may be facing a DoS attach, resulting in port 53 being unavailable.

Security incident report: OS hardening techniques

Project description

Investigate, identify, document and recommend a solution to a website-based security problem by reviewing a tcpdump log.

Section 1: Identify the network protocol involved in the incident
The TCP protocol, at the transport layer of the TCP/IP model, was used to connect the device to the web server. The HTTP protocol, at the application layer of the TCP/IP model, was used after the TCP connection had been established.

Section 2: Document the incident
The Security Team was alerted to the incident by our client who had been receiving complaints from customers. Customer complaints noted that after accessing the yummyrecipesforme.com website they were prompted to download some recipes. After the download had completed, the customers noted that they had been redirected to a new website, greatrecipesforme.com. Customers then noticed that their computer performance had become impaired. The Security Team began the investigation by opening a sandbox and using the packet sniffing tool, tcpdump, to analyze the packet data when attempting to access the yummyrecipesforme.com URL. The log data from tcpdump shows that the TCP protocol, via port 52444, was used to successfully resolve the DNS enquiry. The TCP protocol, via port 36086, was then used to establish the connection between device and web server. The [S], [S.] and [.] flags show the three-way TCP handshake was successfully completed. After successful connection, the team proceeded to download the file, just as the customers had. [P.] flag in the log data indicates a data push, which corresponds with the download request. Following the download request, the log data, at 14:20:32, shows a new DNS request was sent for the greatrecipesforme.com domain. The DNS request is successful and at 14:25.29 our device begins a TCP connection with the web server. The connection is successful and our device then automatically initiates a data push request which the web server acknowledges. We suspected the greatrecipesforme.com domain contained some malicious code to initiate the download of malware onto the host device, this was confirmed by a senior security engineer after reviewing the source code for greatrecipesforme.com. The engineer also noted that the source code for yummyrecipesforme.com had been modified to request the malicious download from visitors.

Section 3: Recommend one remediation for brute force attacks
It is most likely that the admin account for yummyrecipesforme.com was hacked through a brute force attack. The attacker correctly guessed the password and gained access to the admin account. Once the attacker had access, they were able to modify the source code. The following is advised to decrease the risk of future brute force attacks: stronger password requirements, MFA, monitor login attempts, limit the number of login attempts, prohibit the use of previous passwords and more frequent password changes.

Security risk assessment report: network hardening

Project description

Review an organization's overall security posture, following a major data breach, and recommend some network hardening tools.

Part 1: Select up to three hardening tools and methods to implement
MFA Network access privileges Port Filtering

Part 2: Explain your recommendations
MFA improves data confidentiality. MFA will force employees to authorize their login attempt with their smart device/security key – this will limit the practice of shared logins. Network Access Privileges will limit the number of people able to access the database → lower attack surface. Firewall port filtering will also lower the attack surface by only allowing traffic for specific ports through to the network.

Manage file permissions in Linux

Project description

Using Linux bash shell commands to manage file permissions.

Check file and directory details

`cd /home/researcher2/projects` is used to navigate to the target directory.

```
researcher2@3d16540f3ddb:~$ cd /home/researcher2/projects
researcher2@3d16540f3ddb:~/projects$
```

`ls -la` is used to show permissions for all files and directories within the target directory, including any hidden files.

```
researcher2@3d16540f3ddb:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jul  5 09:46 .
drwxr-xr-x 3 researcher2 research_team 4096 Jul  5 10:10 ..
-rw--w---- 1 researcher2 research_team   46 Jul  5 09:46 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jul  5 09:46 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Jul  5 09:46 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jul  5 09:46 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul  5 09:46 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul  5 09:46 project_t.txt
researcher2@3d16540f3ddb:~/projects$
```

Describe the permissions string

The permissions string is a 10-character string.

The first character denotes the file type:

- d = directory
- - = regular file

Characters 2-4 denote the permissions for the user owner type:

- Character 2:
  - r = user has read permissions
  - - = user does not have read permissions
- Character 3:
  - w = user has write permissions
  - - = user does not have write permissions
- Character 4:
  - x = user has execute permissions
  - - = user does not have execute permissions

Characters 5-7 denote the permissions for the group owner type and characters 8-10 denote the permissions for the other owner type; following the same scheme has detailed above for the user owner type.

Using the `project_t.txt` file has an example, we can see that this is a regular file type, the user has read and write permissions, the group also has read and write permissions and other only has read permissions.

Change file permissions

The organization in this example does not want other to have write permissions on files. We can see that the `project_k.txt` file does currently have write permissions for other owner type. To change this we use the `chmod o-w project_k.txt` command:

```
researcher2@3d16540f3ddb:~/projects$ chmod o-w project_k.txt
researcher2@3d16540f3ddb:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jul  5 09:46 .
drwxr-xr-x 3 researcher2 research_team 4096 Jul  5 10:10 ..
-rw--w---- 1 researcher2 research_team   46 Jul  5 09:46 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jul  5 09:46 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jul  5 09:46 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jul  5 09:46 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul  5 09:46 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul  5 09:46 project_t.txt
researcher2@3d16540f3ddb:~/projects$
```

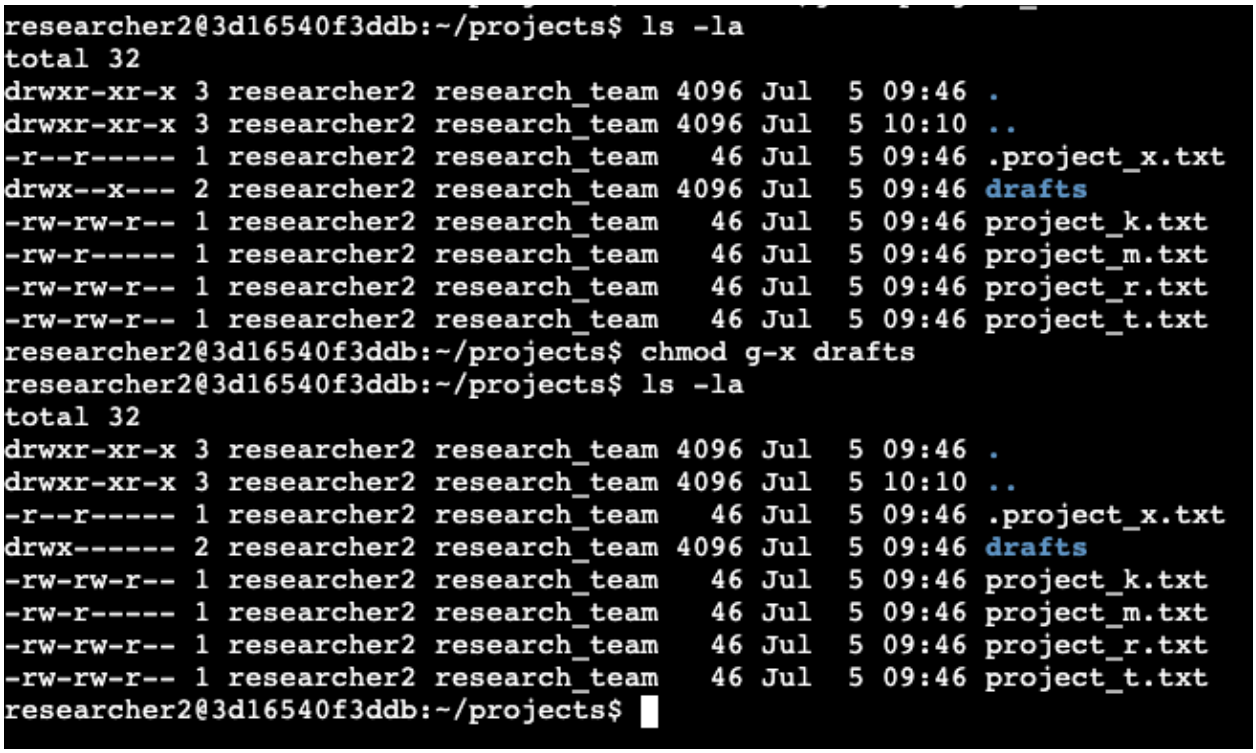
Change file permissions on a hidden file

The hidden file, `project_x.txt` should only include read permission for the user and group. We use the `chmod u=r,g=r project_x.txt` command to achieve this:

```
researcher2@3d16540f3ddb:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jul  5 09:46 .
drwxr-xr-x 3 researcher2 research_team 4096 Jul  5 10:10 ..
-rw--w---- 1 researcher2 research_team   46 Jul  5 09:46 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jul  5 09:46 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jul  5 09:46 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jul  5 09:46 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul  5 09:46 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul  5 09:46 project_t.txt
researcher2@3d16540f3ddb:~/projects$ chmod u=r,g=r .project_x.txt
researcher2@3d16540f3ddb:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jul  5 09:46 .
drwxr-xr-x 3 researcher2 research_team 4096 Jul  5 10:10 ..
-r--r----- 1 researcher2 research_team   46 Jul  5 09:46 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jul  5 09:46 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jul  5 09:46 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jul  5 09:46 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul  5 09:46 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul  5 09:46 project_t.txt
researcher2@3d16540f3ddb:~/projects$
```

Change directory permissions

The `drafts` directory is currently executable by both the user and group, only the user should have access to this directory. To change these permissions we use the `chmod g-x drafts` command:



Summary

In this activity we examined Linux file permissions as well as the commands we can use to view and change file permissions.

Apply filters to SQL queries

Project description

Using SQL operators to filter database queries.

Retrieve after hours failed login attempts

```
MariaDB [organization]> clear
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 (0.003 sec)
```

The above query returns all columns from the `log_in_attempts` table where the login attempt occurred after 18:00 and (making use of the `AND` operator) where the login attempt was unsuccessful (failed).

Retrieve 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 |
|       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 |
|       38 | sbaelish | 2022-05-09 | 14:40:01   | USA     | 192.168.60.42   |       1 |
|       39 | yappiah  | 2022-05-09 | 07:56:40   | MEXICO  | 192.168.57.115  |       1 |
|       42 | cgriffin | 2022-05-09 | 23:04:05   | US      | 192.168.4.157   |       0 |
|       43 | mcouliba | 2022-05-08 | 02:35:34   | CANADA  | 192.168.16.208  |       0 |
|       44 | daquino  | 2022-05-08 | 07:02:35   | CANADA  | 192.168.168.144 |       0 |
|       47 | dkot     | 2022-05-08 | 05:06:45   | US      | 192.168.233.24  |       1 |
```

The above query returns all columns from the `log_in_attempts` table where the login date is either 2022-05-09 or (making use of the `OR` operator) 2022-05-08.

Retrieve 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 |
|       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 |
|       19 | jhill    | 2022-05-12 | 13:09:04   | US      | 192.168.142.245 |       1 |
|       21 | iuduike  | 2022-05-11 | 17:50:00   | US      | 192.168.131.147 |       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 |
|       29 | bisles   | 2022-05-11 | 01:21:22   | US      | 192.168.85.186  |       0 |
|       31 | acook    | 2022-05-12 | 17:36:45   | CANADA  | 192.168.58.232  |       0 |
|       32 | acook    | 2022-05-09 | 02:52:02   | CANADA  | 192.168.142.239 |       0 |
|       33 | zbernal  | 2022-05-11 | 02:52:10   | US      | 192.168.72.59   |       1 |
|       34 | drosas   | 2022-05-11 | 21:02:04   | US      | 192.168.45.93   |       0 |
```

The `NOT` and `LIKE` operators in the above example are used to exclude any value in the country column which starts with the characters MEX. This will filter out both MEX and MEXICO.

Retrieve 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 |
+-------------+--------------+----------+------------+----------+
7 rows in set (0.001 sec)
```

The above query returns employees from the Marketing department (`department = 'Marketing'`) who's offices are located within the East wing of the building (`AND office LIKE 'East%'`).

Retrieve 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 | 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   |
|        1049 | NULL        | jreckley | Finance    | Central-295 |
```

The above query returns employees from the Finance or Sales department (`department = 'Finance' OR department = 'Sales'`).

Retrieve all employees not in IT

```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE 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 | l748m120n401 | drosas   | Sales           | South-292   |
|        1015 | p611q262r945 | jsoto    | Finance         | North-271   |
|        1016 | q793r736s288 | sbaelish | Human Resources | North-229   |
```

Alternate version of the same query using the NOT operator:

```
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 | l748m120n401 | drosas   | Sales           | South-292   |
|        1015 | p611q262r945 | jsoto    | Finance         | North-271   |
|        1016 | q793r736s288 | sbaelish | Human Resources | North-229   |
```

Summary

This activity focused on using SQL operators such as AND, OR and NOT to filter database queries.

Data leak worksheet

Project description

Review the results of a data risk assessment and determine whether effective data handling processes are being implemented.

Incident summary: A sales manager shared access to a folder of internal-only documents with their team during a meeting. The folder contained files associated with a new product that has not been publicly announced. It also included customer analytics and promotional materials. After the meeting, the manager did not revoke access to the internal folder but warned the team to wait for approval before sharing the promotional materials with others.

During a video call with a business partner, a member of the sales team forgot the warning from their manager. The sales representative intended to share a link to the promotional materials so that the business partner could circulate the materials to their customers. However, the sales representative accidentally shared a link to the internal folder instead. Later, the business partner posted the link on their company's social media page assuming that it was the promotional materials.

Control	**Least privilege**
Issue(s)	Lack of technical controls over the internal folder (access rights). Sales team member should not have shared this data with an external party.
Review	NIST SP 800-53 is a customizable information privacy plan. Section AC-6 specifically deals with the principle of least privilege.
Recommendation(s)	Access to sensitive information should be restricted by user role. Access granted should be automatically revoked after a set period. User privileges should also be audited on a regular basis.
Justification	Damage because of data leaks can be mitigated by enforcing strict access control policies. If the correct file rights had been set, the business partner would not have had access to the file data.

Access controls worksheet

Project description

Assess the access controls used by a business following a suspicious payment made by the business to an unknown bank account.

	Note(s)	Issue(s)	Recommendation(s)
Authorization /authentication	**Objective:** Make 1-2 notes of information that can help identify the threat: • The incident occurred on 10/03/2023 at 08:29:57. • User is from the Legal/Admin department. • IP: 152.207.255.255 and computer: Up2-NoGud.	**Objective:** Based on your notes, list 1-2 authorization issues: • User has Admin system rights • Contract termination date was 27/12/2019, however the user accessed their account on 10/03/2023.	**Objective:** Make at least 1 recommendation that could prevent this kind of incident: • Principle of least privilege should be applied. Currently all employees have the same authorization status – Admin. Employees should have role specific access. Authorization permissions should also be audited regularly to ensure employees are deprovisioned appropriately.

Vulnerability Assessment Report

Project description

Conduct a vulnerability assessment for a small business. Evaluate the risks of their information system and outline a remediation plan.

System Description

The server hardware consists of a powerful CPU processor and 128GB of memory. It runs on the latest version of Linux operating system and hosts a MySQL database management system. It is configured with a stable network connection using IPv4 addresses and interacts with other servers on the network. Security measures include SSL/TLS encrypted connections.

Scope

The scope of this vulnerability assessment relates to the current access controls of the system. The assessment will cover a period of three months, from June 2023 to August 2023. NIST SP 800-30 Rev. 1 is used to guide the risk analysis of the information system.

Purpose

The database server contains large amounts of data, including data which is used to assist company employees in finding new customers. It is important for the company to secure the data on the server as it is a critical asset for marketing efforts. The server may also contain Personally Identifiable Information which needs to be securely stored.

Risk Assessment

Threat source	Threat event	Likelihood	Severity	Risk
E.g. Competitor	*Obtain sensitive information via exfiltration*	*3*	*3*	*9*
Employee/Customer (disgruntled)	Obtain or modify sensitive information. Threat could also install malicious software on database and infect other users when they access the database.	3	3	9
Hardware failure	No mention of back-up systems. If there is only one server, this would be a single point of failure. A hardware failure could lead to temporary or permanent data loss.	2	3	6

Approach

The database is currently open to the public, raising concerns over who would potentially access our database. Competitors and disgruntled employees/customers may seek to steal or modify the data on our server. There is no mention of any backups systems – this is a concern as the database has a single point of failure.
Market evidence and expert judgement were considered when deriving the risk scores for each threat source.

Remediation Strategy

Authentication and authorization access control systems are of key importance. Access to data should be role-based and password policies and multi-factor authentication should be implemented. Firewalls should be set-up and implemented so only allowed devices are able to connect to our internal networks (allow access based on approved MAC addresses). All sensitive data should be encrypted and hashed to ensure confidentiality and integrity. Redundancy should also be built into the system to ensure there is no single point of failure: backup power, servers and ISP providers.

Incident handler's journal

Project description

Use an incident handler’s journal to document five different cybersecurity events.

Date: 16/07/2024	**Entry:** 1
Description	Documenting a ransomware attack which used phishing as the attack vector. The ransomware encrypted critical data, forcing the company to shut down their computer systems and suspend their operations. The target was a small healthcare company.
Tool(s) used	None
The 5 W's	Capture the 5 W's of an incident. ● **Who** caused the incident? Unethical hackers ● **What** happened? Phishing email with malicious attachment containing ransomware which encrypted the organizations computer files. ● **When** did the incident occur? Today at 09:00 ● **Where** did the incident happen? Organization offices ● **Why** did the incident happen? Ransome request.
Additional notes	Does the company have backup systems which have not been affected? Can infected systems be wiped and restored with backup data?

Date: 19/07/2024	**Entry:** 2
Description	Investigate a suspicious hash file.
Tool(s) used	VirusTotal
The 5 W's	Capture the 5 W's of an incident. ● **Who** caused the incident? Employee ● **What** happened? Employee downloaded a suspicious file with the SHA-256 file hash of 54e6ea47eb04634d3e87fd7787e2136ccfbcc80ade34f246a12cf93bab527f6b. ● **When** did the incident occur? 1:11pm ● **Where** did the incident happen? Employee's computer – corporate office space ● **Why** did the incident happen? Unknown attacker – possible goal: data manipulation/exfiltration.
Additional notes	What is the goal of the attacker? Is the attack contained to the single infected device? Has any data manipulation/exfiltration occurred?

Date: 19/07/2024	**Entry:** 3
Description	Unauthorized data exfiltration

Tool(s) used	Final Report
The 5 W's	Capture the 5 W's of an incident. ● **Who** caused the incident? Unauthorized hacker ● **What** happened? Web application vulnerability -> forced browsing attack. ● **When** did the incident occur? 22/12/2022 at 15:13 ● **Where** did the incident happen? e-commerce web application ● **Why** did the incident happen? Ransome.
Additional notes	Employee received an initial email from the attacker, but assume it was spam – should the priority level of these emails be reassessed?

Date: 19/07/2024	**Entry:** 4
Description	Monitor network traffic
Tool(s) used	Suricata
The 5 W's	Capture the 5 W's of an incident. **N/A** ● **Who** caused the incident? ● **What** happened? ● **When** did the incident occur? ● **Where** did the incident happen?

	• **Why** did the incident happen?
Additional notes	• Suricata uses the Linux CLI • Rules have 3 components: action, header and rule options • Common actions: alert, drop, pass and reject • Header details the network traffic signature: protocol, source and destination IP and port, traffic direction. • Rule options allow for signature customization. • eve.json files contain more data than the fast.log files. • jq tool is useful for processing JSON data

---

Date: 20/07/2024	**Entry:** 5
Description	Examine failed SSH login from the root mail server account.
Tool(s) used	Splunk
The 5 W's	Capture the 5 W's of an incident. • **Who** caused the incident? Currently unknown – multiple different IP addresses involved • **What** happened? Numerous failed SSH login attempts to access the mail server. >300 between 27/02/2023 and 06/03/2023 • **When** did the incident occur? Between 27/02/2023 and 06/03/2023

	• **Where** did the incident happen? Root account for Buttercup Games mail server • **Why** did the incident happen? Motive unknown
Additional notes	Who do all the failed attempts belong to? What is their objective?

Algorithm for file updates in Python

Project description

Use the Python programming language to create an algorithm which will automate the process of updating a file which contains a list of allowed IP addresses. The algorithm will load and parse through a file containing the current allowed IP addresses and compare those to a list of IP address which should have their access removed.

Open the file that contains the allow list

```
# Assign `import_file` to the name of the file
import_file = "allow_list.txt"
# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.
remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]
# First line of `with` statement
with open(import_file, "r") as file:
```

The file `allow_list.txt` is assigned to the variable `import_file`.
The with statement is used in conjunction with the open function for exception handling and resource management, it will ensure the file is closed at the end of the statement.
The `open()` function is used to open a file and return a `file` object. The file we wish to open is given as an argument to the function (`import_file`). The second argument, `"r"`, is used to indicate that the file should be opened for the purpose of reading the file. The opened `import_file` object is saved to the local variable `file`.

Read the file contents

```
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# Build `with` statement to read in the initial contents of the file

with open(import_file, "r") as file:

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Display `ip_addresses`

print(ip_addresses)
```

```
ip_address
192.168.25.60
192.168.205.12
192.168.97.225
192.168.6.9
192.168.52.90
192.168.158.170
192.168.90.124
192.168.186.176
192.168.133.188
192.168.203.198
192.168.201.40
192.168.218.219
192.168.52.37
192.168.156.224
192.168.60.153
192.168.58.57
192.168.69.116
```

The `.read()` method is used to read the content of the file saved to the `file` variable.
The output of the `.read()` method is saved to a new variable, `ip_addresses`.
The `print()` function is used to output the content of the `ip_addresses` variable.

Convert the string into a list

```
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# Build `with` statement to read in the initial contents of the file

with open(import_file, "r") as file:

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()

# Display `ip_addresses`

print(ip_addresses)
```

```
['ip_address', '192.168.25.60', '192.168.205.12', '192.168.97.225', '192.168.6.9', '192.168.52.90', '192.168.158.17
0', '192.168.90.124', '192.168.186.176', '192.168.133.188', '192.168.203.198', '192.168.201.40', '192.168.218.219',
'192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.58.57', '192.168.69.116']
```

The `.split()` method is used to convert a string into a list. As no argument has been given to the method, a space will be used to differentiate one list item from the next.

Iterate through the IP address list

```
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# Build `with` statement to read in the initial contents of the file

with open(import_file, "r") as file:

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()

# Build iterative statement
# Name loop variable `element`
# Loop through `ip_addresses`

for element in ip_addresses:

    # Display `element` in every iteration

    print(element)
```

```
ip_address
192.168.25.60
192.168.205.12
192.168.97.225
192.168.6.9
192.168.52.90
192.168.158.170
192.168.90.124
```

With the `ip_addresses` variable now of the type list, a for loop is used to iterate from the list of IP addresses. On each iteration, the list item is assigned to the `element` variable. The `print()` function is used within the body of the for loop to output each of the list items.

Remove IP addresses that are on the remove list

```
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# Build `with` statement to read in the initial contents of the file

with open(import_file, "r") as file:

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()

# Build iterative statement
# Name loop variable `element`
# Loop through `ip_addresses`

for element in ip_addresses:

  # Build conditional statement
  # If current element is in `remove_list`,

    if element in remove_list:

        # then current element should be removed from `ip_addresses`

        ip_addresses.remove(element)

# Display `ip_addresses`

print(ip_addresses)
```

```
['ip_address', '192.168.25.60', '192.168.205.12', '192.168.6.9', '192.168.52.90', '192.168.90.124', '192.168.186.17
6', '192.168.133.188', '192.168.203.198', '192.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153',
'192.168.69.116']
```

An `if` statement is add to the body of the for loop to check if the current IP address, `element`, matches any of the IP addresses in the `remove_list` variable. If the condition evaluates to True, then the body of the `if` statement will execute.
In the body of the `if` statement, the `.remove()` method is used with `element` given as the argument. This method will remove the first instant of the IP address associated with the `element` variable from the `ip_addresses` variable.

Update the file with the revised list of IP addresses

```
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# Build `with` statement to read in the initial contents of the file

with open(import_file, "r") as file:

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()

# Build iterative statement
# Name loop variable `element`
# Loop through `ip_addresses`

for element in ip_addresses:

  # Build conditional statement
  # If current element is in `remove_list`,

    if element in remove_list:

        # then current element should be removed from `ip_addresses`

        ip_addresses.remove(element)

# Convert `ip_addresses` back to a string so that it can be written into the text file

ip_addresses = " ".join(ip_addresses)
```

```
# Build `with` statement to rewrite the original file

with open(import_file, "w") as file:

  # Rewrite the file, replacing its contents with `ip_addresses`

  file.write(ip_addresses)

# Build `with` statement to read in the updated file

with open(import_file, "r") as file:

    # Read in the updated file and store the contents in `text`

    text = file.read()

# Display the contents of `text`

print(text)
```

```
ip_address 192.168.25.60 192.168.205.12 192.168.6.9 192.168.52.90 192.168.90.124 192.168.186.176 192.168.133.188 19
2.168.203.198 192.168.218.219 192.168.52.37 192.168.156.224 192.168.60.153 192.168.69.116
```

The `.join()` method is used with the space character to join all the `ip_addresses` list items into a string.
The `with` statement is used in conjunction with the `open()` function to open the `import_file` variable. `"w"` is used as the second argument to the `open()` function, this indicates that we would like to write to the `import_file`. The `.write()` method is used with the `ip_addresses` variable given as input argument; this will overwrite the original IP address data with the updated data.
The file is then opened again and the `.read()` method and output is saved to the `text` variable.
The `print()` function is then used to output the content of the `text` variable.

Summary

This algorithm makes use the following Python elements:

- `with` statements for exception handling and resource management
- `open()` function to both system files in both read and write modes.
- `.read()` method to save the content of a file to a new variable
- `.split()` method to convert from string to list data types
- `for` loop to iterate through list items
- `if` statement to compare list items
- `.remove()` method to remove the first occurrence of an element from a given list
- `.join()` method to combine list items into a single string
- `.write()` method to overwrite existing file data with new data.

The above algorithm could be integrated into a new function which could be reused. The newly defined function could take the current IP address file as well as a list for IP addresses to be removed as parameters to the function:

```
def update_file(import_file, remove_list):

  # Build `with` statement to read in the initial contents of the file

  with open(import_file, "r") as file:

    # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

    ip_addresses = file.read()

  # Use `.split()` to convert `ip_addresses` from a string to a list

  ip_addresses = ip_addresses.split()

  # Build iterative statement
  # Name loop variable `element`
  # Loop through `ip_addresses`

  for element in ip_addresses:

    # Build conditional statement
    # If current element is in `remove_list`,

    if element in remove_list:

      # then current element should be removed from `ip_addresses`

      ip_addresses.remove(element)

  # Convert `ip_addresses` back to a string so that it can be written into the text file

  ip_addresses = " ".join(ip_addresses)

  # Build `with` statement to rewrite the original file

  with open(import_file, "w") as file:

    # Rewrite the file, replacing its contents with `ip_addresses`

    file.write(ip_addresses)
```