SQL para Ciberseguridad: Guía Detallada con Explicaciones Exhaustivas para Estudiantes

Introducción

Este documento está diseñado para estudiantes de SQL que desean aplicar sus conocimientos al campo de la ciberseguridad. Aprenderás a utilizar SQL para analizar datos de seguridad, detectar amenazas y responder a incidentes. Cada consulta se explica en detalle, paso a paso, para que comprendas el propósito de cada cláusula y función.

I. Fundamentos Avanzados de SQL para Ciberseguridad (con Explicaciones Detalladas)

1. **Funciones de Ventana (Window Functions):**
    - **Descripción:** Las funciones de ventana realizan cálculos sobre un conjunto de filas relacionadas con la fila actual, sin agrupar los resultados. Son ideales para análisis de tendencias y detección de anomalías.

Ejemplo: Calcular el promedio móvil de intentos de inicio de sesión fallidos por usuario:

```
SELECT
    username,
    login_date,
    failed_attempts,
    AVG(failed_attempts) OVER (PARTITION BY username ORDER BY login_date ASC ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS rolling_average
FROM
    (SELECT username, login_date, COUNT(*) AS failed_attempts FROM log_in_attempts WHERE success = 0 GROUP BY username, login_date) AS subquery
ORDER BY
    username,
```

- login_date;
-
    - **Explicación Detallada:**
        - `SELECT username, login_date, failed_attempts`: Selecciona el nombre de usuario, la fecha de inicio de sesión y el número de intentos fallidos.
        - `AVG(failed_attempts) OVER (PARTITION BY username ORDER BY login_date ASC ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)`: Calcula el promedio móvil de intentos fallidos.
            - `AVG(failed_attempts)`: Calcula el promedio de la columna `failed_attempts`.
            - `OVER (PARTITION BY username)`: Divide los datos en particiones basadas en el nombre de usuario. El cálculo del promedio se realiza por separado para cada usuario.

- `ORDER BY login_date ASC`: Ordena los datos dentro de cada partición por fecha de inicio de sesión en orden ascendente.
- `ROWS BETWEEN 6 PRECEDING AND CURRENT ROW`: Define la ventana de filas para el cálculo del promedio. Incluye las 6 filas anteriores a la fila actual y la fila actual.

- `FROM (SELECT username, login_date, COUNT(*) AS failed_attempts FROM log_in_attempts WHERE success = 0 GROUP BY username, login_date) AS subquery`: Crea una subconsulta para obtener el número de intentos fallidos por usuario y fecha.
  - `SELECT username, login_date, COUNT(*) AS failed_attempts`: Selecciona el nombre de usuario, la fecha de inicio de sesión y cuenta el número de intentos fallidos.
  - `FROM log_in_attempts WHERE success = 0`: Selecciona los registros de la tabla `log_in_attempts` donde el inicio de sesión falló (`success = 0`).
  - `GROUP BY username, login_date`: Agrupa los resultados por nombre de usuario y fecha de inicio de sesión.

- `ORDER BY username, login_date`: Ordena los resultados por nombre de usuario y fecha de inicio de sesión.

- **Análisis:** Esta consulta calcula el promedio móvil de intentos fallidos de inicio de sesión para cada usuario durante los últimos 7 días. Un aumento significativo en el promedio móvil podría indicar un ataque de fuerza bruta.

2.
3. **Expresiones de Tabla Comunes (Common Table Expressions - CTEs):**
   - **Descripción:** Las CTEs permiten definir consultas temporales que se pueden utilizar dentro de una consulta principal. Mejoran la legibilidad y la modularidad de las consultas complejas.

Ejemplo: Identificar usuarios que han iniciado sesión desde múltiples ubicaciones en un corto período de tiempo:

```
WITH
  LoginLocations AS (
```

```
    SELECT
      username,
      COUNT(DISTINCT ip_address) AS location_count
    FROM
      log_in_attempts
    WHERE
      login_date BETWEEN DATE('now', '-1 day') AND DATE('now')
    GROUP BY
      username
  )
SELECT
  username,
  location_count
FROM
  LoginLocations
WHERE
  location_count > 3
ORDER BY
```

- location_count DESC;
-
  - **Explicación Detallada:**
    - `WITH LoginLocations AS (...)`: Define una CTE llamada `LoginLocations`.
    - `SELECT username, COUNT(DISTINCT ip_address) AS location_count`: Selecciona el nombre de usuario y cuenta el número de direcciones IP distintas desde las que ha iniciado sesión.
    - `FROM log_in_attempts WHERE login_date BETWEEN DATE('now', '-1 day') AND DATE('now')`: Selecciona los registros de la tabla `log_in_attempts` donde la fecha de inicio de sesión está dentro del último día.
    - `GROUP BY username`: Agrupa los resultados por nombre de usuario.
    - `SELECT username, location_count FROM LoginLocations WHERE location_count > 3`: Selecciona el nombre de usuario y el número de ubicaciones de la CTE `LoginLocations` donde el número de ubicaciones es mayor que 3.
    - `ORDER BY location_count DESC`: Ordena los resultados por número de ubicaciones en orden descendente.
  -
  - **Análisis:** Esta consulta utiliza una CTE para contar el número de ubicaciones distintas desde las que cada usuario ha iniciado sesión en el

último día. Los usuarios que han iniciado sesión desde más de 3 ubicaciones podrían ser objeto de una investigación más profunda.

-
4.
5. **Funciones de Cadena Avanzadas:**
    - **Descripción:** Las funciones de cadena permiten manipular y analizar datos de texto. Son útiles para extraer información de registros de eventos y detectar patrones maliciosos.

Ejemplo: Extraer dominios de URLs en registros de navegación web:

```
SELECT
  url,
  SUBSTR(url, INSTR(url, '//') + 2, INSTR(SUBSTR(url, INSTR(url, '//') + 2), '/') - 1) AS domain
FROM
  web_access_logs
WHERE
```

- url LIKE '%//%';
-
    - **Explicación Detallada:**
        - `SELECT url, SUBSTR(url, INSTR(url, '//') + 2, INSTR(SUBSTR(url, INSTR(url, '//') + 2), '/') - 1) AS domain`: Selecciona la URL y extrae el dominio.
            - `SUBSTR(url, start, length)`: Extrae una subcadena de la URL.
            - `INSTR(url, '//') + 2`: Encuentra la posición de "//" en la URL y suma 2 para obtener la posición del primer carácter del dominio.
            - `INSTR(SUBSTR(url, INSTR(url, '//') + 2), '/') - 1`: Encuentra la posición de "/" en la subcadena que comienza después de "//" y resta 1 para obtener la longitud del dominio.
        -
        - `FROM web_access_logs WHERE url LIKE '%//%'`: Selecciona los registros de la tabla `web_access_logs` donde la URL contiene "//".
    -
    - **Análisis:** Esta consulta extrae el dominio de cada URL en los registros de acceso web. Esto puede ayudar a identificar sitios web maliciosos o sospechosos a los que los usuarios están accediendo.
-
6.

7. **Funciones de Fecha y Hora Avanzadas:**
    - **Descripción:** Las funciones de fecha y hora permiten manipular y analizar datos de tiempo. Son útiles para detectar anomalías temporales y patrones de comportamiento inusuales.

Ejemplo: Identificar intentos de inicio de sesión fuera del horario laboral normal:

```
SELECT
  username,
  login_time
FROM
  log_in_attempts
WHERE
  STRFTIME('%H', login_time) NOT BETWEEN '08' AND '17'
ORDER BY
```

    - login_time DESC;
    -
        - **Explicación Detallada:**
            - `SELECT username, login_time`: Selecciona el nombre de usuario y la hora de inicio de sesión.
            - `FROM log_in_attempts WHERE STRFTIME('%H', login_time) NOT BETWEEN '08' AND '17'`: Selecciona los registros de la tabla `log_in_attempts` donde la hora de inicio de sesión no está entre las 8 AM y las 5 PM.
                - `STRFTIME('%H', login_time)`: Extrae la hora de la columna `login_time` en formato de 24 horas.
                - `NOT BETWEEN '08' AND '17'`: Verifica que la hora no esté entre las 8 AM y las 5 PM.
            -
            - `ORDER BY login_time DESC`: Ordena los resultados por hora de inicio de sesión en orden descendente.
        -
        - **Análisis:** Esta consulta identifica los intentos de inicio de sesión que ocurrieron fuera del horario laboral normal (8 AM a 5 PM). Esto podría indicar un acceso no autorizado o un comportamiento sospechoso.
    -
8.

II. Técnicas Avanzadas de Análisis de Seguridad con SQL (con Explicaciones Detalladas)

1. **Análisis de Comportamiento Basado en Perfiles:**
    - **Descripción:** Crear perfiles de comportamiento para usuarios, sistemas o redes y detectar desviaciones de estos perfiles.

Ejemplo: Crear un perfil de acceso a recursos para un usuario y detectar accesos inusuales:

```
-- Crear un perfil de acceso a recursos para un usuario
CREATE TABLE user_resource_profile AS
SELECT
    username,
    resource_accessed,
    COUNT(*) AS access_count
FROM
    access_logs
WHERE
    username = 'usuario_especifico'
GROUP BY
    username,
    resource_accessed;

-- Detectar accesos inusuales
SELECT
    a.username,
    a.resource_accessed,
    a.access_time
FROM
    access_logs a
LEFT JOIN
    user_resource_profile p ON a.username = p.username AND a.resource_accessed =
p.resource_accessed
WHERE
    a.username = 'usuario_especifico'
    AND p.resource_accessed IS NULL
ORDER BY
```

- a.access_time DESC;
-
  - **Explicación Detallada:**
    - `CREATE TABLE user_resource_profile AS ...`: Crea una tabla llamada `user_resource_profile` con los resultados de la consulta.
    - `SELECT username, resource_accessed, COUNT(*) AS access_count`: Selecciona el nombre de usuario, el recurso accedido y cuenta el número de accesos.
    - `FROM access_logs WHERE username = 'usuario_especifico'`: Selecciona los registros de la tabla `access_logs` donde el nombre de usuario es 'usuario_especifico'.
    - `GROUP BY username, resource_accessed`: Agrupa los resultados por nombre de usuario y recurso accedido.

- `SELECT a.username, a.resource_accessed, a.access_time`: Selecciona el nombre de usuario, el recurso accedido y la hora de acceso de la tabla `access_logs`.
- `FROM access_logs a LEFT JOIN user_resource_profile p ON a.username = p.username AND a.resource_accessed = p.resource_accessed`: Realiza una unión izquierda entre la tabla `access_logs` y la tabla `user_resource_profile` basada en el nombre de usuario y el recurso accedido.
- `WHERE a.username = 'usuario_especifico' AND p.resource_accessed IS NULL`: Selecciona los registros donde el nombre de usuario es 'usuario_especifico' y el recurso accedido no está en la tabla `user_resource_profile`.
- `ORDER BY a.access_time DESC`: Ordena los resultados por hora de acceso en orden descendente.

- **Análisis:** Esta técnica crea un perfil de acceso a recursos para un usuario específico y luego detecta los accesos a recursos que no están en su perfil. Esto podría indicar un comportamiento anómalo o un intento de acceso no autorizado.

2.
3. **Análisis de Anomalías Basado en Estadísticas:**
   - **Descripción:** Utilizar técnicas estadísticas para identificar valores atípicos en los datos de seguridad.

Ejemplo: Detectar transferencias de datos inusualmente grandes utilizando la desviación estándar:

```
SELECT
    source_ip,
    destination_ip,
    bytes_transferred,
    timestamp
FROM
    network_traffic_logs
WHERE
    bytes_transferred > (SELECT AVG(bytes_transferred) + (3 * STDDEV(bytes_transferred)) FROM network_traffic_logs)
ORDER BY
```

   - bytes_transferred DESC;

   - **Explicación Detallada:**

- `SELECT source_ip, destination_ip, bytes_transferred, timestamp`: Selecciona la dirección IP de origen, la dirección IP de destino, el número de bytes transferidos y la marca de tiempo.
- `FROM network_traffic_logs WHERE bytes_transferred > (SELECT AVG(bytes_transferred) + (3 * STDDEV(bytes_transferred)) FROM network_traffic_logs)`: Selecciona los registros de la tabla `network_traffic_logs` donde el número de bytes transferidos es mayor que el promedio más 3 veces la desviación estándar.
  - `AVG(bytes_transferred)`: Calcula el promedio del número de bytes transferidos.
  - `STDDEV(bytes_transferred)`: Calcula la desviación estándar del número de bytes transferidos.
-
- `ORDER BY bytes_transferred DESC`: Ordena los resultados por número de bytes transferidos en orden descendente.

-
- **Análisis:** Esta consulta identifica las transferencias de datos que son más de 3 desviaciones estándar por encima del promedio. Esto podría indicar un intento de exfiltración de datos o un comportamiento anómalo.

4.
5. **Análisis de Grafos:**
   - **Descripción:** Utilizar bases de datos de grafos y consultas SQL para analizar las relaciones entre entidades de seguridad (usuarios, sistemas, redes, etc.).

Ejemplo: Identificar usuarios que están conectados a sistemas comprometidos:

```
-- Asumiendo que tienes una base de datos de grafos con nodos para usuarios y sistemas, y aristas para conexiones
SELECT
    u.name AS username,
    s.name AS system_name
FROM
    users u
JOIN
    connections c ON u.id = c.user_id
JOIN
    systems s ON c.system_id = s.id
WHERE
```

   - s.compromised = TRUE;
   -
     - **Explicación Detallada:**

- `SELECT u.name AS username, s.name AS system_name`: Selecciona el nombre de usuario de la tabla `users` y el nombre del sistema de la tabla `systems`.
- `FROM users u JOIN connections c ON u.id = c.user_id JOIN systems s ON c.system_id = s.id`: Realiza una unión entre las tablas `users`, `connections` y `systems` basada en las relaciones entre usuarios, conexiones y sistemas.
- `WHERE s.compromised = TRUE`: Selecciona los registros donde el sistema está comprometido.

- **Análisis:** Esta consulta identifica a los usuarios que están conectados a sistemas que han sido comprometidos. Esto puede ayudar a determinar el alcance del compromiso y a tomar medidas para proteger otros sistemas.

6.

III. Casos de Uso Avanzados y Ejemplos Prácticos (con Explicaciones Detalladas)

1. **Detección de Ataques de Phishing:**
   - **Escenario:** Identificar correos electrónicos de phishing que están llegando a los usuarios.

SQL:

```
SELECT
   sender_email,
   subject,
   body
FROM
   email_logs
WHERE
   subject LIKE '%urgente%'
   AND body LIKE '%haga clic aquí%'
   AND sender_email NOT IN (SELECT email FROM trusted_senders)
ORDER BY
```

   - timestamp DESC;

   - **Explicación Detallada:**
     - `SELECT sender_email, subject, body`: Selecciona el correo electrónico del remitente, el asunto y el cuerpo del correo electrónico.
     - `FROM email_logs WHERE subject LIKE '%urgente%' AND body LIKE '%haga clic aquí%' AND sender_email NOT IN`

`(SELECT email FROM trusted_senders)`: Selecciona los registros de la tabla `email_logs` donde el asunto contiene la palabra "urgente", el cuerpo contiene la frase "haga clic aquí" y el correo electrónico del remitente no está en la tabla `trusted_senders`.

- `subject LIKE '%urgente%'`: Verifica que el asunto contenga la palabra "urgente".
- `body LIKE '%haga clic aquí%'`: Verifica que el cuerpo contenga la frase "haga clic aquí".
- `sender_email NOT IN (SELECT email FROM trusted_senders)`: Verifica que el correo electrónico del remitente no esté en la tabla `trusted_senders`.

- `ORDER BY timestamp DESC`: Ordena los resultados por marca de tiempo en orden descendente.

- **Análisis:** Esta consulta identifica los correos electrónicos que tienen un asunto urgente, un cuerpo que contiene un enlace para hacer clic y un remitente que no está en la lista de remitentes de confianza. Esto podría indicar un correo electrónico de phishing.

2.
3. **Análisis de Vulnerabilidades:**
   - **Escenario:** Identificar sistemas que tienen vulnerabilidades conocidas.

SQL:

```
SELECT
  s.name AS system_name,
  v.name AS vulnerability_name,
  v.severity
FROM
  systems s
JOIN
  vulnerabilities v ON s.os = v.os AND s.version = v.version
WHERE
  v.severity = 'critical'
ORDER BY
```

s.name;

- **Explicación Detallada:**
  - `SELECT s.name AS system_name, v.name AS vulnerability_name, v.severity`: Selecciona el nombre del

sistema de la tabla `systems`, el nombre de la vulnerabilidad de la tabla `vulnerabilities` y la gravedad de la vulnerabilidad.

- `FROM systems s JOIN vulnerabilities v ON s.os = v.os AND s.version = v.version`: Realiza una unión entre la tabla `systems` y la tabla `vulnerabilities` basada en el sistema operativo y la versión.
- `WHERE v.severity = 'critical'`: Selecciona los registros donde la gravedad de la vulnerabilidad es "critical".
- `ORDER BY s.name`: Ordena los resultados por nombre del sistema.

- **Análisis:** Esta consulta identifica los sistemas que tienen vulnerabilidades críticas conocidas. Esto puede ayudar a priorizar la aplicación de parches y otras medidas de seguridad.

4.

5. **Detección de Malware:**
   - **Escenario:** Identificar sistemas que están infectados con malware.

SQL:

```
SELECT
   s.name AS system_name,
   m.name AS malware_name,
   m.detection_time
FROM
   systems s
JOIN
   malware_detections m ON s.id = m.system_id
ORDER BY
```

   - m.detection_time DESC;

   - **Explicación Detallada:**
     - `SELECT s.name AS system_name, m.name AS malware_name, m.detection_time`: Selecciona el nombre del sistema de la tabla `systems`, el nombre del malware de la tabla `malware_detections` y la hora de detección.
     - `FROM systems s JOIN malware_detections m ON s.id = m.system_id`: Realiza una unión entre la tabla `systems` y la tabla `malware_detections` basada en el ID del sistema.
     - `ORDER BY m.detection_time DESC`: Ordena los resultados por hora de detección en orden descendente.

        -
        - **Análisis:** Esta consulta identifica los sistemas que han sido detectados con malware. Esto puede ayudar a tomar medidas para aislar y limpiar los sistemas infectados.
    -
6.

IV. Optimización de Consultas SQL para Ciberseguridad (con Explicaciones Detalladas)

1. **Indexación:**
    - **Descripción:** Crear índices en las columnas que se utilizan con frecuencia en las cláusulas WHERE y JOIN para acelerar las consultas.
    - **Ejemplo:** Crear un índice en la columna `username` de la tabla `log_in_attempts`:
      CREATE INDEX idx_username ON log_in_attempts (username);
    -
        - **Explicación Detallada:**
            - `CREATE INDEX idx_username ON log_in_attempts (username)`: Crea un índice llamado `idx_username` en la columna `username` de la tabla `log_in_attempts`. Esto permite que las consultas que filtran por nombre de usuario se ejecuten más rápido.
        -
    -
2.
3. **Particionamiento:**
    - **Descripción:** Dividir las tablas grandes en particiones más pequeñas para mejorar el rendimiento de las consultas.
    - **Ejemplo:** Particionar la tabla `log_in_attempts` por fecha:
      CREATE TABLE log_in_attempts_2023 PARTITION OF log_in_attempts FOR VALUES FROM ('2023-01-01') TO ('2023-12-31');
    -
        - **Explicación Detallada:**
            - `CREATE TABLE log_in_attempts_2023 PARTITION OF log_in_attempts FOR VALUES FROM ('2023-01-01') TO ('2023-12-31')`: Crea una tabla llamada `log_in_attempts_2023` que es una partición de la tabla `log_in_attempts` para los valores de fecha entre el 1 de enero de 2023 y el 31 de diciembre de 2023. Esto permite que las consultas que filtran por fecha se ejecuten más rápido.

        -
    -
4.
5. **Utilización de Vistas Materializadas:**
    - **Descripción:** Crear vistas materializadas para almacenar los resultados de consultas complejas que se ejecutan con frecuencia.

Ejemplo: Crear una vista materializada para almacenar el número de intentos de inicio de sesión fallidos por usuario:

```
CREATE MATERIALIZED VIEW failed_login_counts AS
SELECT
  username,
  COUNT(*) AS failed_attempts
FROM
  log_in_attempts
WHERE
  success = 0
GROUP BY
```

    - username;
    -
        - **Explicación Detallada:**
            - `CREATE MATERIALIZED VIEW failed_login_counts AS ...`: Crea una vista materializada llamada `failed_login_counts` con los resultados de la consulta.
            - `SELECT username, COUNT(*) AS failed_attempts`: Selecciona el nombre de usuario y cuenta el número de intentos fallidos.
            - `FROM log_in_attempts WHERE success = 0`: Selecciona los registros de la tabla `log_in_attempts` donde el inicio de sesión falló.
            - `GROUP BY username`: Agrupa los resultados por nombre de usuario.
        -
    -
6.

V. Herramientas y Técnicas Adicionales

1. **SIEM (Security Information and Event Management):** Integrar consultas SQL en SIEMs para automatizar la detección de amenazas y la respuesta a incidentes.
2. **Threat Intelligence:** Utilizar fuentes de inteligencia de amenazas para enriquecer los datos de seguridad y mejorar la precisión de las consultas SQL.

3. **Machine Learning:** Utilizar técnicas de aprendizaje automático para detectar anomalías y patrones complejos en los datos de seguridad.

VI. Consideraciones de Seguridad

1. **Inyección SQL:** Proteger las bases de datos de seguridad contra ataques de inyección SQL.
2. **Control de Acceso:** Limitar el acceso a las bases de datos de seguridad solo a los analistas autorizados.
3. **Auditoría:** Habilitar la auditoría de las consultas SQL para rastrear las acciones de los analistas y detectar posibles abusos.

Conclusión

SQL es una herramienta esencial para los analistas de ciberseguridad. Al dominar las técnicas avanzadas de SQL y comprender cómo aplicarlas a casos de uso específicos, los analistas pueden mejorar significativamente su capacidad para detectar amenazas, responder a incidentes y proteger los activos de una organización. La integración de SQL con otras herramientas y técnicas de seguridad puede potenciar aún más la eficacia del análisis de seguridad.