SIMATS SCHOOL OF ENGINEERING

SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

CHENNAI-602105

Detecting Data leaks using SQL

A CAPSTONE PROJECT REPORT

Submitted in the partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCES

Submitted by

(192211311) Swarna Sai Venkata Vamsi Kousik

Under the Supervision of

Dr.K.Anbazhagan

DECLARATION

I,Swarna Sai Venkata Vamsi Kousik students of'**Bachelor of Engineering in Computer Sciences**, Department of Computer Science and Engineering, SIMATS, Saveetha University, Chennai, hereby we declare that the work presented in this Capstone Project Work entitled **"Detecting Data leaks using SQL"** is the outcome of our Bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics.

192211311 Swarna Sai Venkata Vamsi Kousik

Date:29-07-2024

Place: SSE

CERTIFICATE

This is to certify that the project entitled **“Detecting Data leaks using SQL”** submitted by , Swarna Sai Venkata Vamsi Kousik has been carried out under our supervision. The project has been submitted as per the requirements in the current semester of B. Tech Information Technology.

Teacher-in-charge

Dr.K.Anbazhagan

TABLE OF CONTENTS

ABSTRACT:

In today's digital landscape, the threat of data leaks poses significant risks to both individuals and organizations. These leaks can occur through various vulnerabilities, including SQL injection attacks, unauthorized access, and inadvertent data exposure. This project proposes a comprehensive system for detecting and mitigating data leaks using SQL-based technologies. The system leverages advanced SQL queries and database monitoring techniques to identify anomalous data access patterns, suspicious queries, and potential vulnerabilities in real-time. By integrating with cloud technologies and employing robust encryption mechanisms such as AESX, the system aims to enhance data security and privacy protection. Through content inspection, contextual analysis, and proactive monitoring, the system not only detects potential breaches but also takes swift corrective actions to prevent data loss. This project ultimately aims to safeguard sensitive information held by e-commerce platforms and other digital environments, ensuring the integrity and confidentiality of user data.This abstract encapsulates the purpose, methods, and goals of the project in detecting and mitigating data leaks using SQL technology

INTRODUCTION:

Detecting data leaks is a critical aspect of cybersecurity, especially in environments where databases store sensitive information. SQL (Structured Query Language) plays a pivotal role in managing and querying these databases. However, SQL-based systems are vulnerable to various threats, such as SQL injection attacks and unauthorized access, which can lead to data breaches.The process of detecting data leaks using SQL involves leveraging the capabilities of SQL queries and database monitoring techniques to identify anomalies and potential vulnerabilities. By analyzing SQL query logs, database activity logs, and network traffic data, organizations can proactively detect suspicious activities indicative of data breaches.Key methodologies include preprocessing collected data to ensure accuracy and consistency, extracting meaningful features from SQL queries and user behaviors, and training models—such as anomaly detection algorithms or rule-based systems—to identify and respond to potential data leaks promptly.In essence, detecting data leaks using SQL combines technical expertise in database management with advanced analytical techniques to safeguard sensitive information and maintain data integrity within organizational infrastructures. This approach is crucial for mitigating risks and enhancing cybersecurity resilience in today's digital landscape.

Key components of the framework include:

- **Data Collection:** Gather SQL query logs, database activity logs, and network traffic data.
- **Preprocessing:** Clean data, handle missing values, and normalize formats.
- **Feature Extraction:** Extract query patterns and analyze user behaviors.
- **Model Training:** Develop anomaly detection models and rule-based systems.
- **Network Analysis:** Monitor network traffic for SQL injection and behavioral anomalies.

In today's interconnected world, the protection of sensitive data is paramount due to the increasing prevalence of cyber threats. Data leaks, whether accidental or malicious, can lead to severe consequences such as financial loss, reputational damage, and legal liabilities. Detecting and mitigating these leaks is a complex but essential task, particularly within databases managed through SQL.SQL, as the standard language for managing and querying relational databases, offers powerful capabilities but also introduces vulnerabilities. Common issues include SQL injection attacks, where malicious code is injected into SQL statements to gain unauthorized access or manipulate data. Effective detection of data leaks using SQL involves a multifaceted approach that encompasses proactive monitoring, anomaly detection, and stringent access controls.

PROBLEM STATEMENT:

1. *Data Collection and Storage:*

- **Database Logs:** Set up database logging to capture SQL queries, transactions, and access events. Use database-specific logging mechanisms or third-party tools that integrate with the database.
- **Transaction History:** Maintain a transaction history table within the database to record details such as user IDs, timestamp, executed queries, tables accessed, and data transferred.

2. *SQL Queries and Analysis:*

- **Real-time Monitoring Queries:** Develop SQL queries or stored procedures to monitor database activities in real-time. Examples include:

  ```
  sql
  Copy code
  SELECT username, query_text, timestamp
  FROM database_log
  WHERE timestamp >= NOW() - INTERVAL 5 MINUTE;
  ```

  This query retrieves recent queries executed within the last 5 minutes.

- **Anomaly Detection Queries:** Implement SQL queries to detect anomalies such as:
  - Sudden spikes in data export:

    ```
    sql
    Copy code
    SELECT username, COUNT(*) AS export_count
    FROM transaction_history
    WHERE action = 'export'
    GROUP BY username
    HAVING export_count > (SELECT AVG(export_count) * 2 FROM (SELECT
    username, COUNT(*) AS export_count FROM transaction_history WHERE
    action = 'export' GROUP BY username) AS avg_export_count);
    ```

    This query identifies users whose data export activities exceed twice the average export count.

  - Unauthorized access to sensitive tables:

```
sql
Copy code
SELECT username, table_name, COUNT(*) AS access_count
FROM transaction_history
WHERE table_name IN ('sensitive_table1', 'sensitive_table2')
AND action = 'query'
AND timestamp >= NOW() - INTERVAL 1 HOUR
GROUP BY username, table_name
HAVING access_count > 0;
```

This query flags users querying sensitive tables within the last hour.

3. Alerting and Notification:

- **Stored Procedures for Alerting:** Create SQL stored procedures to generate alerts based on predefined thresholds or patterns identified by anomaly detection queries.
- **Email/SMS Notifications:** Integrate SQL with email or SMS APIs to send notifications to administrators or security teams when potential data leaks are detected.

4. Reporting and Analysis:

- **Dashboard and Reports:** Develop SQL queries or views to generate reports summarizing detected incidents, trends in database access, and potential vulnerabilities.
- **Scheduled Reports:** Schedule SQL jobs or scripts to automatically generate and distribute reports on a daily, weekly, or monthly basis.

5. Integration and Scalability:

- **SIEM Integration:** Integrate with Security Information and Event Management (SIEM) systems to centralize monitoring, correlation of events, and further analysis.
- **Scalability Considerations:** Ensure SQL queries and scripts are optimized for performance to handle large volumes of database transactions without impacting database responsiveness.

6. Compliance and Audit:

- **Audit Trail:** Maintain an audit trail of SQL queries executed by the detection system for compliance and audit purposes.
- **Regulatory Compliance:** Ensure the design adheres to regulatory requirements (e.g., GDPR, HIPAA) governing data protection and privacy.

7. Testing and Maintenance:

- **Testing:** Conduct thorough testing to validate the effectiveness and accuracy of SQL queries in detecting simulated data leak scenarios.
- **Maintenance:** Regularly review and update SQL queries, stored procedures, and detection rules based on evolving threats and access patterns.

PROPOSED DESIGN:

The proposed design for monitoring suspicious discussions on online forums using data mining techniques involves a multi-layered approach that integrates data collection, preprocessing, feature extraction, model training, and real-time monitoring. The framework leverages natural language processing (NLP), machine learning (ML), and network analysis to effectively identify and analyze suspicious activities.

REQUIREMENT GATHERING AND ANALYSIS:

1. Identify Stakeholders:

- **Security Team:** Responsible for overseeing data security and compliance.
- **Database Administrators:** Manage and maintain the corporate database environment.
- **IT Operations:** Ensure the availability and performance of database systems.
- **Legal and Compliance Officers:** Ensure adherence to regulatory requirements (e.g., GDPR, HIPAA).
- **Business Owners:** Understand data sensitivity and criticality for different business functions.

2. Define Goals and Objectives:

- **Goal:** Develop a system to detect and prevent data leaks within the corporate database environment using SQL.
- **Objectives:**
  - Real-time monitoring of database activities.
  - Detection of unauthorized access or suspicious data transfers.
  - Timely alerts to security teams for rapid response.
  - Compliance with regulatory standards and internal policies.

3. Gather Functional Requirements:

- **Real-time Monitoring:**
  - Capture SQL queries, transactions, and access events in real-time.
  - Monitor database logs for query execution and data transfer activities.
- **Anomaly Detection:**
  - Identify anomalies such as unusual spikes in data exports or unauthorized access to sensitive tables.
  - Implement thresholds and patterns for detecting potential data leaks.
- **Alerting Mechanism:**
  - Generate alerts via email, SMS, or integrate with existing incident response platforms.
  - Notify administrators or security teams promptly upon detection of potential data leaks.
- **Reporting and Analysis:**
  - Generate reports summarizing detected incidents, trends in database access, and vulnerabilities.
  - Provide dashboards for real-time monitoring and historical analysis.
- **Integration and Scalability:**
  - Integrate with Security Information and Event Management (SIEM) systems for centralized monitoring and correlation.
  - Ensure scalability to handle large volumes of database transactions without performance degradation.

4. Non-Functional Requirements:

- **Performance:** Ensure SQL queries and scripts are optimized for performance to minimize impact on database operations.
- **Security:** Implement strong access controls and encryption mechanisms to protect sensitive data.

- **Compliance:** Adhere to regulatory requirements (e.g., GDPR, HIPAA) governing data protection and privacy.
- **Auditability:** Maintain audit trails of SQL queries and detection activities for compliance and audit purposes.

5. Constraints and Assumptions:

- **Technology Constraints:** Use SQL-based solutions compatible with the corporate database systems (e.g., MySQL, PostgreSQL, SQL Server).
- **Budget and Resource Constraints:** Consider limitations on budget, personnel, and infrastructure for implementing and maintaining the detection system.
- **Assumptions:** Assume availability of database logging mechanisms and necessary permissions to access and analyze database logs.

6. Risk Assessment:

- **Data Sensitivity:** Assess the sensitivity and criticality of different types of data stored in the databases.
- **Security Risks:** Identify potential risks associated with data leaks, such as reputational damage, financial loss, or regulatory penalties.
- **Operational Risks:** Consider risks related to system performance, scalability, and integration with existing IT infrastructure.

7. Prioritization and Feasibility:

- **Prioritize Requirements:** Rank requirements based on importance to security, compliance, and business needs.
- **Feasibility Assessment:** Evaluate the technical feasibility of implementing SQL-based solutions for real-time monitoring, anomaly detection, and alerting.

8. Documentation:

- **Requirements Document:** Document all gathered requirements, including functional and non-functional requirements, constraints, assumptions, and risk assessments.
- **Design Specifications:** Prepare detailed specifications for the SQL queries, stored procedures, and data structures to be implemented.

By following these steps, you can effectively gather and analyze requirements for developing a SQL-based system to detect data leaks within a corporate database environment. This structured approach ensures alignment with stakeholder needs, compliance with regulatory standards, and effective mitigation of security risks associated with data breaches.

TOOL SELECTION CRITERIA:

- **Compatibility and Integration:**
  - Ensure compatibility with your database systems (e.g., MySQL, PostgreSQL).
  - Ability to integrate with existing database logging mechanisms.
- **Real-time Monitoring:**
  - Capabilities for real-time streaming and processing of database logs.
  - Low latency to detect and respond to data leaks promptly.
- **Anomaly Detection:**
  - Advanced analytics for detecting unusual access patterns and behaviors.
  - Customizable rules for defining thresholds and detection criteria.
- **Alerting and Notification:**
  - Robust alerting mechanisms with configurable notifications (e.g., email, SMS).
- **Reporting and Visualization:**
  - Customizable dashboards and detailed reports on incidents and trends.
- **Scalability and Performance:**
  - Ability to scale to handle large volumes of database transactions.
  - Optimized performance to minimize impact on database operations.
- **Security and Compliance:**
  - Strong data security measures (encryption, access controls).
  - Compliance with data protection regulations (e.g., GDPR, HIPAA).
- **Ease of Use and Management:**
  - User-friendly interface for setup, configuration, and monitoring.
  - Manageable administrative overhead for ongoing operations.
- **Vendor Support and Community:**
  - Reputation of vendor for support, updates, and responsiveness.
  - Availability of community resources for additional support and knowledge sharing.
- **Cost and Licensing:**
  - Total cost of ownership including initial setup, licensing, and maintenance.
  - Alignment of licensing model with budget and scalability requirements.

SCANNING AND TESTING METHODOLOGIES:

1. Database Vulnerability Scanning:

- **Purpose:** Identify known vulnerabilities and misconfigurations in the database environment that could lead to data leaks.
- **Methodology:**
    - Utilize automated vulnerability scanning tools specific to database management systems (e.g., Nessus, OpenVAS).
    - Scan for common issues such as weak authentication mechanisms, improper access controls, outdated software versions, and configuration errors.
- **Outcome:**
    - Generate reports detailing identified vulnerabilities and their severity levels.
    - Prioritize and remediate vulnerabilities to prevent potential data leaks.

2. SQL Injection Testing:

- **Purpose:** Detect and prevent SQL injection vulnerabilities, a common attack vector that can lead to unauthorized access and data leaks.
- **Methodology:**
    - Perform penetration testing using automated tools (e.g., SQLMap, Burp Suite) or manual testing techniques.
    - Craft malicious SQL queries to exploit vulnerabilities and gain unauthorized access to sensitive data.
    - Test input validation mechanisms and parameterized queries to prevent SQL injection attacks.
- **Outcome:**
    - Identify vulnerable SQL injection points and potential paths for data leakage.
    - Implement secure coding practices and security controls to mitigate SQL injection risks.

3. Database Activity Monitoring:

- **Purpose:** Monitor and analyze database activities in real-time to detect suspicious or unauthorized access patterns indicative of data leaks.
- **Methodology:**
    - Deploy database activity monitoring tools or solutions capable of capturing and analyzing SQL queries, transactions, and access logs.
    - Define baseline behaviors and set thresholds for detecting anomalies (e.g., unusual data export volumes, access to sensitive tables).
    - Implement alerting mechanisms to notify administrators of potential data leaks promptly.
- **Outcome:**
    - Detect and respond to unauthorized database activities and potential data leaks in real-time.
    - Investigate incidents and implement corrective actions to strengthen database security.

4. Data Loss Prevention (DLP) Testing:

- **Purpose:** Validate the effectiveness of DLP policies and controls in preventing unauthorized data transfers and leaks.
- **Methodology:**
    - Conduct testing scenarios to simulate data exfiltration attempts, such as exporting sensitive data or accessing restricted information.

    - Evaluate how DLP solutions detect and prevent unauthorized data transfers initiated through SQL queries.
    - Test policy enforcement mechanisms and response capabilities to incidents.
- **Outcome:**
    - Assess the robustness of DLP controls in mitigating data leakage risks.
    - Adjust and optimize DLP policies based on testing results and findings.

5. Compliance and Audit Testing:

- **Purpose:** Ensure adherence to regulatory requirements (e.g., GDPR, HIPAA) and internal security policies related to data protection and privacy.
- **Methodology:**
    - Conduct compliance audits focusing on SQL-based data handling practices, access controls, and encryption standards.
    - Verify that SQL queries and database configurations comply with data protection regulations and organizational policies.
    - Review audit logs and documentation to demonstrate compliance with legal and regulatory requirements.
- **Outcome:**
    - Identify non-compliance issues and gaps in data protection practices.
    - Implement corrective measures to address findings and enhance data leak prevention measures.

USER AUTHENTICATION AND ROLE-BASED ACCESS CONTROL.

User Authentication

User authentication ensures that only authorized users can access the database system and perform operations. It forms the foundation for secure data access and helps in preventing unauthorized data leaks.

1. **Implementation:**
    - Utilize strong authentication mechanisms such as username/password combinations or multifactor authentication (MFA) where possible.
    - Implement database-specific authentication mechanisms provided by SQL-based systems (e.g., MySQL, PostgreSQL).
2. **Best Practices:**
    - Enforce password policies (e.g., complexity, expiration) to strengthen authentication security.
    - Regularly audit user accounts and remove inactive or unnecessary accounts to reduce attack surface.
    - Monitor and log authentication attempts to detect and respond to suspicious activities.

Role-Based Access Control (RBAC)

RBAC ensures that users have appropriate permissions based on their roles within the organization. It helps enforce the principle of least privilege, limiting access to data and functionalities necessary for users to perform their duties.

1. **Implementation:**
    - Define roles based on job functions or responsibilities (e.g., admin, analyst, developer).

- Assign permissions to roles rather than individual users to streamline access management.
- Use SQL commands to create roles and grant/revoke permissions accordingly:

```
sql
Copy code
CREATE ROLE analyst;
GRANT SELECT ON sensitive_table TO analyst;
```

2. **Best Practices:**
   - Regularly review and update role assignments to align with organizational changes and least privilege principles.
   - Implement separation of duties to prevent conflicts of interest and reduce the risk of insider threats.
   - Audit and monitor role assignments and permissions to ensure compliance with security policies and regulatory requirements.

Detecting Data Leaks Using Authentication and RBAC

1. **Monitoring Database Access:**
   - Log and monitor user authentication events to detect unauthorized access attempts or anomalies.
   - Implement SQL queries to analyze authentication logs for unusual patterns (e.g., failed login attempts, access from unusual locations).
2. **Auditing Data Access:**
   - Use SQL auditing mechanisms or triggers to capture data access events (e.g., SELECT, INSERT, UPDATE, DELETE).
   - Monitor access to sensitive data tables and review audit logs regularly for unauthorized queries or data exports.
3. **Alerting and Incident Response:**
   - Configure alerting mechanisms based on predefined thresholds or suspicious activities detected through authentication and access logs.
   - Implement SQL scripts or stored procedures to generate alerts and notifications to security teams for immediate response.
4. **Compliance and Reporting:**
   - Generate compliance reports detailing user access patterns, authentication events, and adherence to RBAC policies.
   - Include audit trails and documentation of access controls implemented through SQL-based mechanisms for regulatory compliance audits.

UI DESIGN:

1. Dashboard Overview:

- **Overview Widgets:** Include widgets summarizing key metrics such as total queries executed, data export volume, and active user sessions.
- **Graphical Representations:** Use charts (e.g., line charts, bar charts) to visualize trends in database activities over time, such as query frequency or data transfer size.

2. Real-time Monitoring:

- **Active Alerts Panel:** Display a panel showing active alerts and their severity levels, indicating potential data leak incidents.
- **Recent Events Feed:** Provide a feed displaying real-time database events, such as user logins, query executions, and data exports.

3. Anomaly Detection:

- **Anomaly Visualization:** Highlight anomalies detected, such as sudden spikes in query volume or unusual data export patterns, using color-coded alerts or trend lines on charts.
- **Threshold Indicators:** Show thresholds and deviations from normal activity levels to draw attention to potential data leaks.

4. Incident Management:

- **Alert Details:** Clicking on an alert should provide detailed information about the incident, including affected database tables, user responsible, and actions taken.
- **Incident Response Actions:** Include options to escalate incidents, initiate investigations, and apply remediation actions directly from the UI.

5. User Management:

- **User Roles and Permissions:** Provide a section for managing user roles and permissions within the UI, ensuring secure access control management.
- **Audit Logs:** Access logs of user activities related to data leak detection and management to support compliance and accountability.

6. Reporting and Analysis:

- **Customizable Reports:** Offer options to generate customizable reports on detected incidents, trends in database access, and compliance with security policies.
- **Export Functionality:** Allow exporting reports in various formats (e.g., PDF, CSV) for further analysis and regulatory reporting.

7. Configuration and Settings:

- **Alert Settings:** Enable administrators to configure alert thresholds, notification preferences (e.g., email, SMS), and escalation rules.
- **Integration Settings:** Provide options to integrate with other security tools, such as SIEM systems, for centralized monitoring and correlation of events.

8. Accessibility and Usability:

- **Responsive Design:** Ensure the UI is responsive and accessible across different devices (e.g., desktops, tablets, mobile phones).
- **Intuitive Navigation:** Design an intuitive navigation structure with clear menus, breadcrumbs, and search functionalities for easy usability.

9. Security Considerations:

- **Authentication:** Implement strong authentication mechanisms (e.g., MFA) to secure access to the UI.
- **Data Encryption:** Ensure sensitive data displayed within the UI, such as audit logs and incident details, are encrypted both at rest and in transit.

Example Components:

- **Dashboard:** Widgets showing query statistics, data export trends, and active alerts.
- **Charts:** Line charts displaying query frequency and bar charts indicating data export volume.
- **Alerts Panel:** Real-time alerts with severity levels and actionable options.
- **Reports Section:** Customizable reports on incidents and compliance.
- **User Management:** Role-based access controls and audit logs.

CONCLUSION:

Detecting data leaks using SQL is crucial for maintaining the security and integrity of organizational databases. By leveraging SQL's capabilities, organizations can implement a proactive approach to monitor database activities in real-time, including SQL queries, data transfers, and user access. This continuous monitoring helps detect unauthorized access attempts, suspicious behavior, or anomalies such as unusual data export volumes that may indicate potential data leaks. Implementing advanced SQL queries and algorithms enables organizations to identify patterns and deviations from normal database usage, facilitating early detection and response to security incidents. Configuring alerting mechanisms ensures that administrators and security teams receive timely notifications upon detecting potential data leaks, allowing for swift investigation and mitigation actions. Role-Based Access Control (RBAC) further enhances security by restricting database access based on users' roles and responsibilities, thereby minimizing the risk of unauthorized data access and insider threats. Compliance with regulatory requirements is also supported through SQL-based auditing and reporting functionalities, providing organizations with the necessary documentation to demonstrate adherence to data protection regulations. By continuously improving SQL-based detection strategies and security controls, organizations can effectively mitigate risks associated with data leaks, ensuring operational resilience and maintaining stakeholder trust.