Database Security

Course 1

Introduction

Bibliography

1. Afyouni, H., Database Security and Auditing: Protecting Data Integrity and Accessibility, Cengage Learning, 2006.

2. Basta, A., Zgola, M., Database Security, Cengage Learning, 2011

3. Kenan, K., Cryptography in the Database: The Last Line of Defense, Addison Wesley Publishing Company, 2005.

4. Knox, D., Maroulis, W., Gaetjen, S., Oracle Database 12c Security, Oracle Press, 2015.

5. Natan, R., Implementing Database Security and Auditing, Elsevier, 2005.

6. Neagu, A., Oracle 11g Anti-Hacker's Cookbook, Packt Publishing, 2012.

7. Thuraisingham, B., Database and Applications Security: Integrating Information Security and Data Management, Auerbach Publications, 2005.

Contents

1. Intro / Importance of the field

- Growth of the number of reported incidents (loss or unauthorized exposure of data)
- **The quantity** of collected, stored and shared data is continuously growing

=> The need to secure the databases

1. Intro / Importance of the field

- Data breaches appear (very) frequently
- Target companies: from large commercial chains to sports club sites
- How do they happen? What happens with compromised data?
- Database security - impact on the actual information systems' design

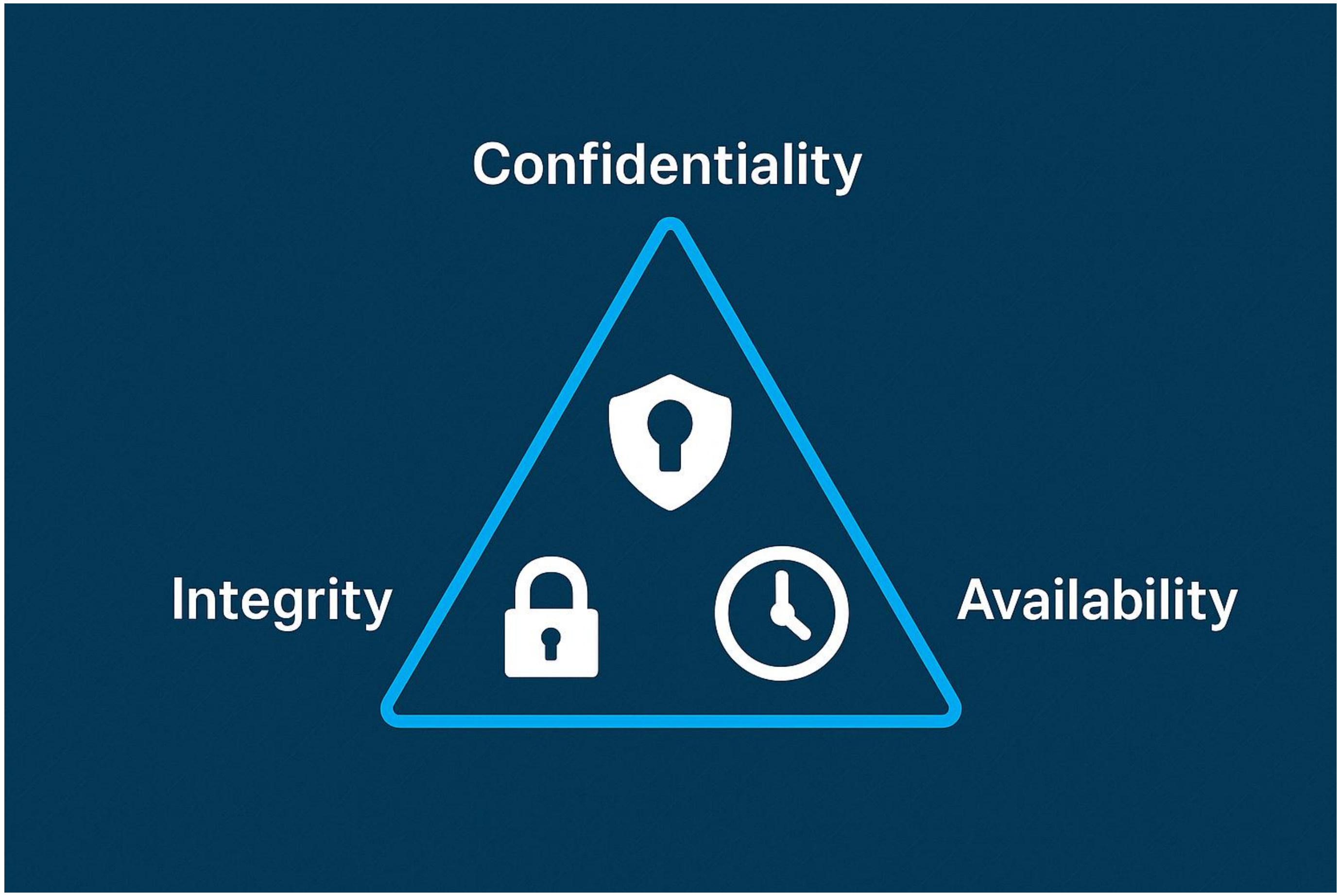
1. Intro / Aim

- Database security should:
  - Ensure controlled and protected access to the database contents
  - Preserve integrity, consistency and quality of the data
- Database security includes/ intersects with a great variety of other topics: physical security, network security, encryption, authentication, authorization etc.

1. Intro / Concepts on data security

- Out of the aspects the database security might include, an important topic is the data security.
- Data security involves:
  - Confidentiality (data protection to be disclosed – accessed in an unauthorized way);
  - Integrity (prevent unauthorized access to data);
  - Availability (identify and recover following hardware or software errors or malicious actions which result in the denial of data availability)

1. Intro / Concepts on data security



1. Intro / Concepts… / Data security

- The 3 previous aspects include, at their turn, the following topics:
  - Access control
  - Applications’ access
  - Vulnerability
  - Inference
  - Audit mechanisms

1. Intro / Concepts… / Data security

- **Access control** is the process by which users and database objects are granted rights and privileges
- **Application access** refers to the necessity to grant adequate access rights to the external applications, requiring a database connection
- **Vulnerability** refers to “weaknesses” which allow malicious users to exploit resources

1. Intro / Concepts… / Data security

- **Inference** refers to the use of legitimate data for the purpose of inferring unknown information, for whose direct retrieval there had not been granted rights
- **Database audit** records the access to the database and the users' activity, providing a way of identifying breaches

1. Intro / Structure of the course

- The course is structured in 2 parts:
  - A **theoretical** part:
    - A (short) history of the Database Security field
    - Theoretical fundamentals
  - A (more) **practical** part:
    - Concepts and techniques on security in relational database management systems (authentication, authorization, data encryption, profiles, privileges, roles, audit etc.), with reference to Oracle system (versions >=11g/12c)
    - Database applications security, data hiding etc.

2. Current state

- The database technology is a component of the core of many systems.
- Databases allow data storage and sharing
- The quantity (volume) of data stored in these systems is growing (exponentially)
- We can say the same about the need to ensure data integrity and to secure access to data.

2. Current state

PAST:

- *The Privacy Rights Clearing House* (2010) reported that more than 345 million client records had been lost or stolen since 2005 (since such incidents have been investigated)
- Ponemon Institute (2009) reported the average cost of a data crime: 202 $ for a client record
- In august 2009 it took place one of the greatest data security crimes: there had been stolen over 130 million debit and credit card numbers using a well-known vulnerability of databases, SQL Injection

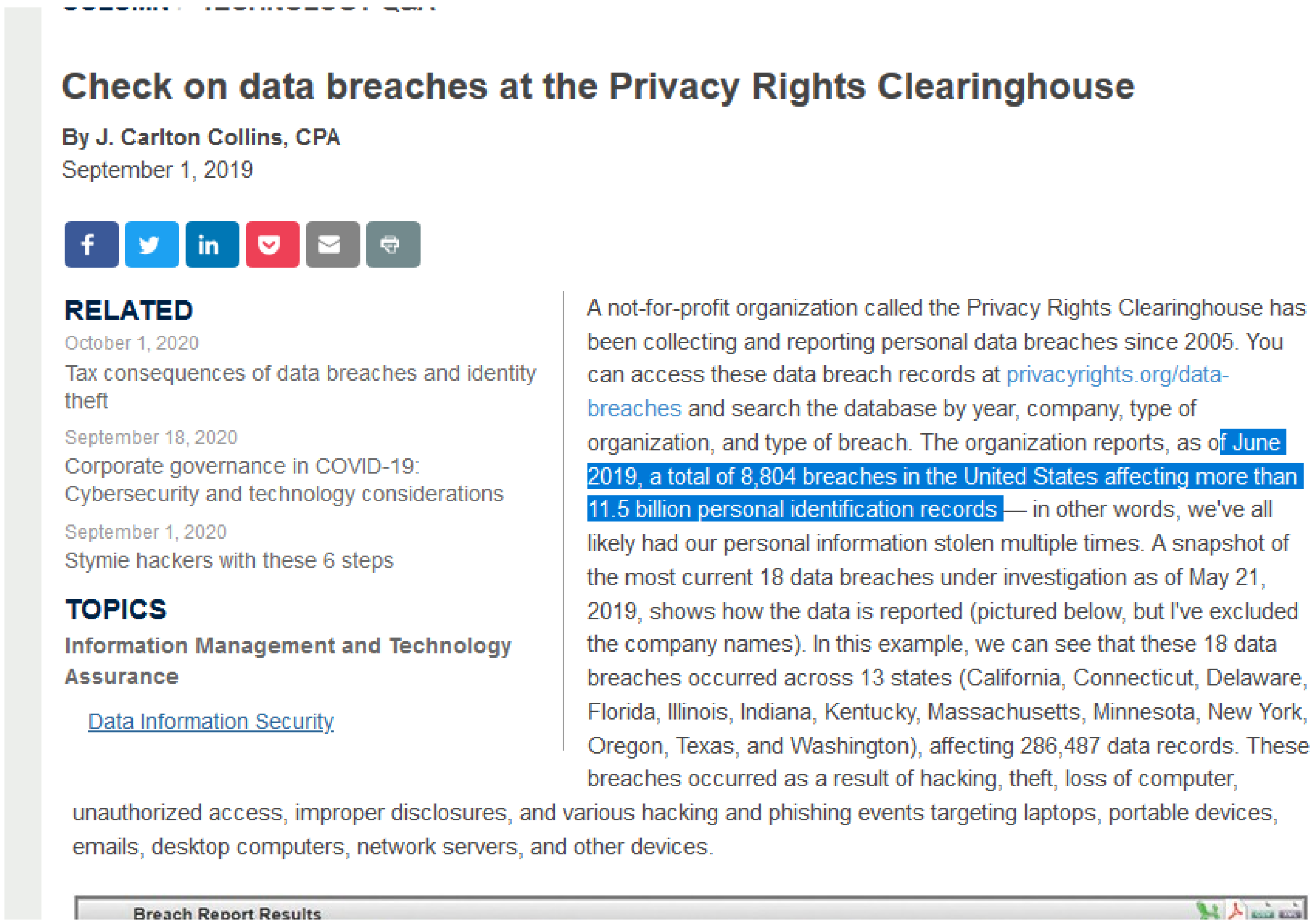
2. Current state

PRESENT:

- 460 million records exposed in may 2020: *https://www.bleepingcomputer.com/news/security/over-460-million-records-exposed-in-breach-incidents-reported-in-may/*
- The database of data security breaches: *https://privacyrights.org/data-breaches*

2. Current state

PRESENT:



Check on data breaches at the Privacy Rights Clearinghouse

By J. Carlton Collins, CPA
September 1, 2019

RELATED
October 1, 2020
Tax consequences of data breaches and identity theft
September 18, 2020
Corporate governance in COVID-19: Cybersecurity and technology considerations
September 1, 2020
Stymie hackers with these 6 steps

TOPICS
Information Management and Technology Assurance
Data Information Security

A not-for-profit organization called the Privacy Rights Clearinghouse has been collecting and reporting personal data breaches since 2005. You can access these data breach records at privacyrights.org/data-breaches and search the database by year, company, type of organization, and type of breach. The organization reports, as of June 2019, a total of 8,804 breaches in the United States affecting more than 11.5 billion personal identification records — in other words, we've all likely had our personal information stolen multiple times. A snapshot of the most current 18 data breaches under investigation as of May 21, 2019, shows how the data is reported (pictured below, but I've excluded the company names). In this example, we can see that these 18 data breaches occurred across 13 states (California, Connecticut, Delaware, Florida, Illinois, Indiana, Kentucky, Massachusetts, Minnesota, New York, Oregon, Texas, and Washington), affecting 286,487 data records. These breaches occurred as a result of hacking, theft, loss of computer, unauthorized access, improper disclosures, and various hacking and phishing events targeting laptops, portable devices, emails, desktop computers, network servers, and other devices.

Breach Report Results

(https://www.journalofaccountancy.com/issues/2019/sep/data-breaches-privacy-rights-clearinghouse.html)

2. Current state

- *Verizon Business Risk Team* reports statistics on data crimes since 2004; they investigated 90 crimes in 2008
- They reported that more than 285 million records had been compromised, and this number overcomes the total of the previous years
- The studies considered who commits such actions and how they appear
  - Most of the crimes appear from external sources: 75% come from outside the organization
  - 91% of the compromised records had a connection with groups of organized crime
  - Most of the crimes took place through hacking and often they were facilitated by errors committed by the victim
  - Unauthorized access and SQL Injection were found as the most common forms of hacking, an interesting fact when one takes into account that these are known and often preventable.

2. Current state

- *Verizon Business Risk Team* – the DBIR (Data Breach Investigations Report):

https://www.verizon.com/business/resources/reports/dbir/

- An analysis of the reports for the years 2020-2023:

https://delinea.com/blog/verizon-2020-dbir-5-top-takeaways

https://delinea.com/blog/verizon-data-breach-investigations-report-top-takeaways

https://resources.infosecinstitute.com/topic/4-key-takeaways-from-the-2022-verizon-dbir-report/

https://delinea.com/blog/2023-verizon-data-breach-investigations-report-insights

https://delinea.com/blog/2024-verizon-dbir-credential-compromise-dominates

https://blog.gitguardian.com/verizon-dbir-2025/

https://www.verizon.com/business/resources/infographics/2025-dbir-infographic.pdf

[Aspects of database security]

- Access control
- Applications’ access
- Database vulnerability
- Inference
- Audit mechanisms

3. Access Control

- The primary method to protect data is to limit access to it
- This can be realized by:
  - Authentication
  - Authorization
  - Access control
- These 3 mechanisms are distinct, but they are usually used together.

3. Access Control

- The access control determines the **granularity** with which the rights are granted to some users and objects
  - The DBMSs use a form of **authentication** (for example, with username and password) in order to restrict the access to the system
  - Users are **authorized** or granted privileges on the resources
  - Access control **refines** this process by granting permissions on the objects (tables, views, rows and columns) and data sets.
- ***Example:*** Student_A can have the right to login on the university database, with authorizations that include the read-only on the table Course_list. By the granular level of access control, the students can consult the list of courses but not the grades received by their colleagues.
- The limitation of the access to the database objects can be realized by the **mechanism of control Grant/Revoke**.

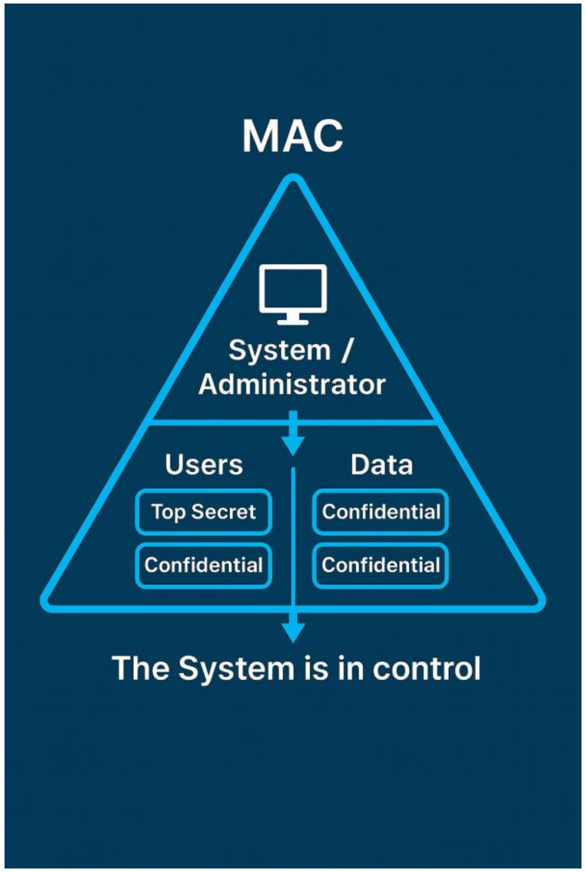
3. Access Control (Grant/Revoke)

- Base concept in security
- Limits the actions (select, insert, update, delete, execute) that the users can perform on objects (tables, columns, views, stored procedures)
- Can be defined in 3 ways:
  - Mandatory Access Control (MAC)
  - Discretionary Access Control (DAC)
  - Role Based Access Control (RBAC)
- ***Example***: Prof. Smith receives read privileges on the table Students

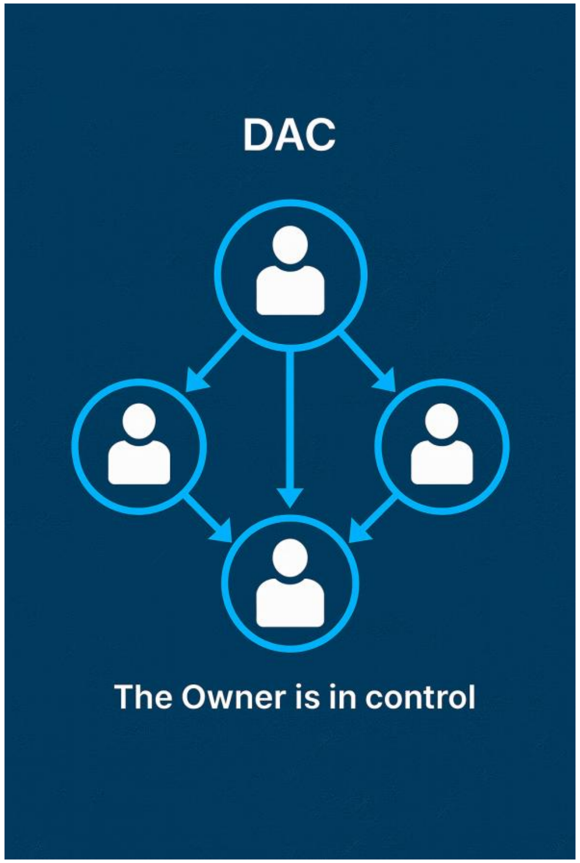
3. Access Control (Grant/Revoke)

- MAC and DAC provide privileges to the users and groups
- The MAC rules are applied at the system's level, they are static and considered safer
  - ***Example***: Prof. Smith received read access on the table Students
- The DAC rules are provided at the user's level, they are dynamic and content-oriented
  - ***Example***: Prof. Smith receives read access on the table Students, but only for the students enrolled to a certain course

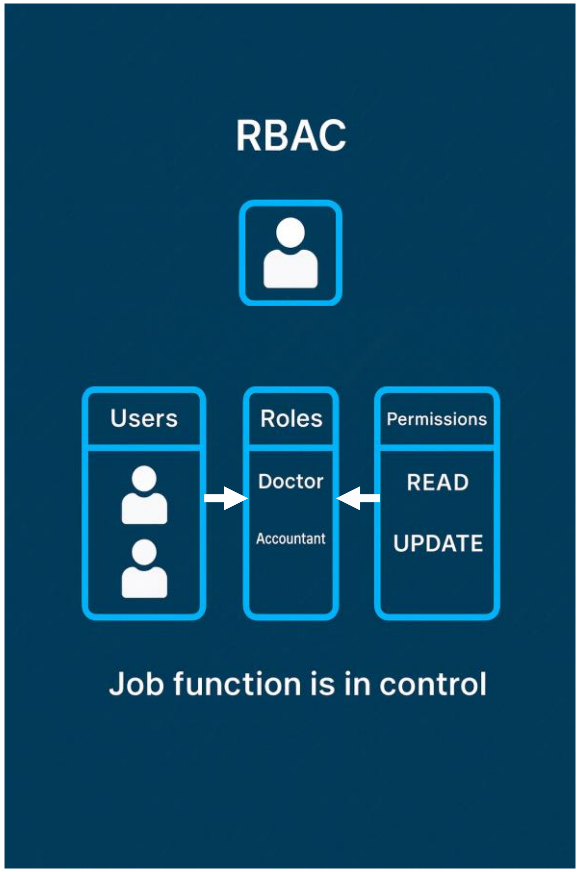
3. Access Control (Grant/Revoke)



The System is in control. Rules are centrally managed and cannot be changed by users.



The Owner is in control. It's flexible but less secure for large systems.



Job function is in control. This is the most efficient model for organizations.

3. Access Control (Grant/Revoke)

- RBAC is efficient for database systems
  - Role = job
  - Assumes the identification of operations and objects on which operations require access
  - The users attached to a role receive automatically the role's privileges
  - ***Example***:
    - Prof. Smith can be attached to a role Faculty
    - The role Faculty contains the right to read the table Students, to get the date of the enrollment to a certain course, to update the grades of the students assigned to the course.

3. Access Control (Grant/Revoke)

- The identification of the users and of their processing and data access needs is an important step in establishing **good protocols for the security of the databases**
- The identification and definition of roles, the correct granting of access rights on the objects and actions, the appropriate association of the users with the declared roles represent the **difficulty** of this process.

3. Access Control (Grant/Revoke)

- After a role is created, the RBAC implementation follows the model:

  GRANT privilege_name
  ON object_name TO role_name;

  - *privilege_name* identifies the rights that can be granted; these can include the data selection, update or alter the database structure
  - *ON* identifies the database objects
  - *TO* identifies the roles on which the privileges are applied
  - **Example**: Prof. Smith was granted the role *Faculty*, and this one was granted read rights on the table Students. The RBAC rule is:

    GRANT select ON students TO faculty;

3. Access Control (Grant/Revoke)

- *REVOKE* – revokes the rights and withdraws the users' authorizations from roles
- **Example**: The removal of privileges on the table Students from the role *Faculty*. => The members of this role will not be able to access the table's data

3. Access Control (Grant/Revoke)

- Syntactically, the role creation and the RBAC implementation are simple
- The challenge consists in the users' and the associated roles' management
- This includes not only the correct identification of the roles, but also the continuous management of the granted ones.
- The general security rule: granting the most restrictive set of privileges that are necessary to perform the authorized tasks = **Principle of least privilege**
- Building the organizational structure of a RBAC system can become **complex**, and the need to frequently change the users' roles means that RBAC requires **constant monitoring**.

3. Access Control (Grant/Revoke)

- In the book *Security Metrics: Replacing Fear, Uncertainty and Doubts* (Jaquith, 2007): "*Today's information security battleground is all about entitlements – who's got them, whether they were defined properly, and how to enforce them*"

3. Access Control / Row-level security

- The control of the access to tables or columns can be ensured through the simple granting of privileges on them
- Restricting access to rows' data needs additional steps
- **Example**: A student should be able to access or modify only the records that are related to himself
- A common way to implement the row-level security is defined with the help of the SQL views

3. Access Control / Row-level security

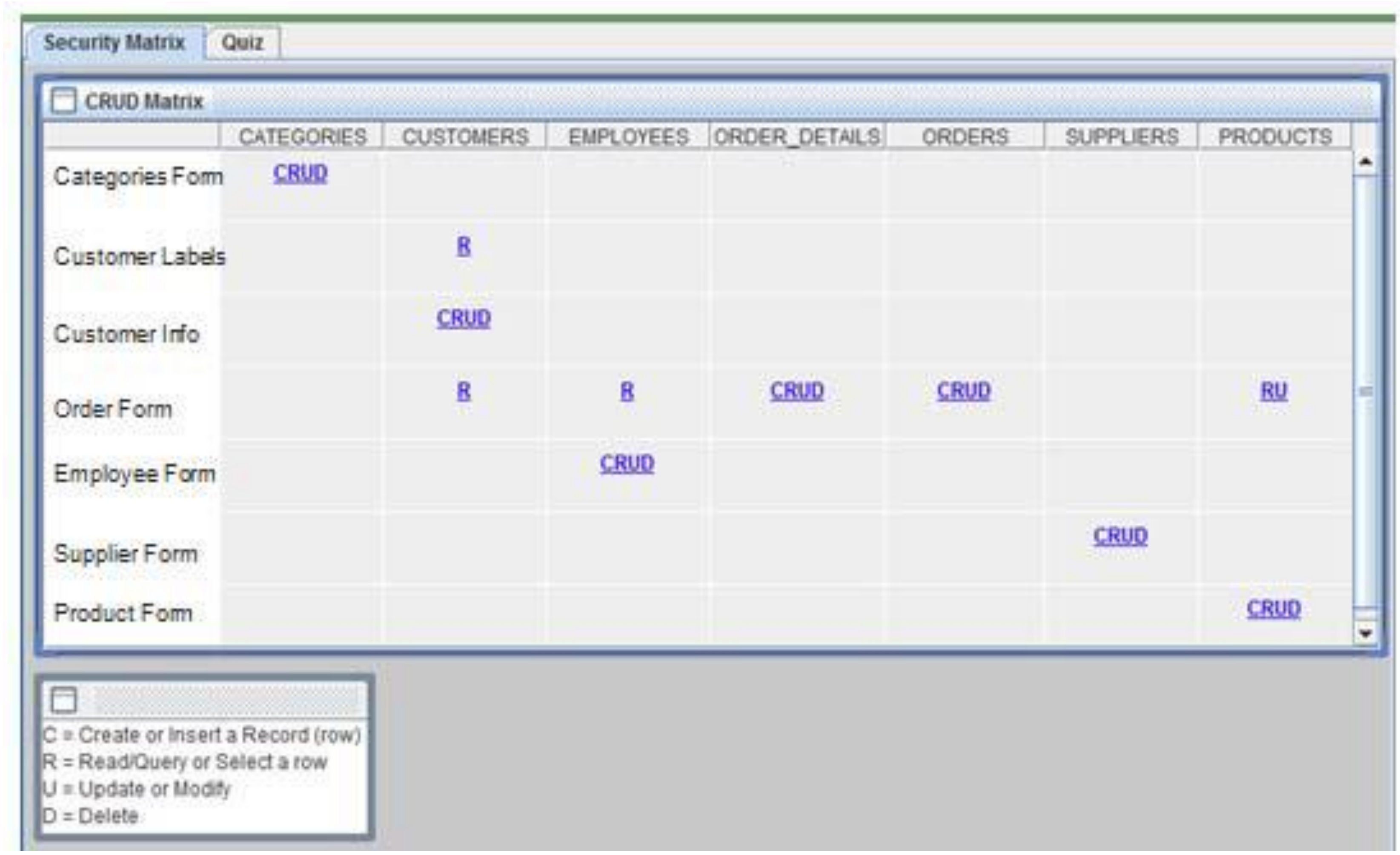
- One can build a view which executes a SELECT statement which returns certain rows, based on certain conditions (for example, related to the current user)
- **Example**:

```
CREATE VIEW view_profile
AS SELECT * FROM app_users
    WHERE username = USER;
```

4. Applications' access

- Many users don't access the database through a direct connection, but rather through an application
- A simple tool, called security matrix (**CRUD**), can be used to explicitly identify the access rights needed by the application
- A visual representation of the correlation between operations or authorizations and the input/output sources (for example, forms and reports)

4. Applications' access



Security Matrix | Quiz

CRUD Matrix

	CATEGORIES	CUSTOMERS	EMPLOYEES	ORDER_DETAILS	ORDERS	SUPPLIERS	PRODUCTS
Categories Form	CRUD						
Customer Labels		R					
Customer Info		CRUD					
Order Form		R	R	CRUD	CRUD		RU
Employee Form			CRUD				
Supplier Form						CRUD	
Product Form							CRUD

C = Create or Insert a Record (row)
R = Read/Query or Select a row
U = Update or Modify
D = Delete

4. Applications' access

- Another advantage of the security matrix: it describes visually the integrity rules
  - Simplifies the identification of all the applications that can be affected by a modification on a database table
  - For example, the removal of a column from the table Products will impact the corresponding forms, raising an error during the execution of these applications
- Before the modification, the impact must be known and, so, the applications that need updates can be determined

5. Databases’ Vulnerability

- The security breaches are a frequently encountered phenomenon
- More and more databases have become accessible through web applications

➢ Their exposure to security threats rises

- The goal is to reduce the vulnerability when facing these threats

5. Databases' Vulnerability

- One of the most known vulnerabilities of database applications is SQL Injection
- SQL Injection includes the issue of risks that are related to unvalidated input
- It happens when SQL statements are created dynamically, based on the user's input
- The threat appears when users input actual code that leads to the execution of unauthorized statements
- The vulnerability appears due to the SQL language features, which allow comments within statements (--), concatenation of SQL statements separated by ; and the capacity to query metadata from the data dictionary
- The **solution**: the validation of input

5. Databases' Vulnerability

- **Example:** Login process on a web page which validates a username and a password based on the data stored in a relational database
- The web page has an authentication form
- The string provided by the user is used for the dynamic creation of a SQL statement which searches records in the database
- A malicious user can provide a string which allows him to gain access to data which is otherwise inaccessible to him
- For example, the following string: ' OR 1=1 - - provided in the authentication form allows access to the system without even knowing a username and a password
- The reason: the application generates a dynamic query resulted from the concatenation of certain strings with the user's input values

5. Databases’ Vulnerability

- **Example**:

```
SELECT count(*) FROM app_users
WHERE username = ‘username_textbox_value’
AND password = ‘password_textbox_value’;
```

- After the input of a username and a password, the query becomes:

```
SELECT count(*) FROM app_users
WHERE username =’user1’ AND password = ‘pass1’;
```

- If a user inputs the string OR 1=1 - -’ the query becomes:

```
SELECT count(*) FROM app_users

WHERE username = ‘’ OR 1 = 1 - - ’

AND password = ‘’;
```

5. Databases' Vulnerability

- The expression 1 = 1 is true for each row of the table, determining the clause OR to return true.
- The characters - - comment the rest of the SQL statement
- The query will return a result greater than 0, which means that at least one row exists in the users' table, leading to a successful login

5. Databases' Vulnerability

- Another type of SQL Injection appears when the system allows the processing of stacked queries. These imply the execution of more than one query with a single function call inside an application.
- **Example**: Initially, the user has the permission to select the attributes of the products from the table Products. The user injects a stacked query that embeds an additional SQL query. The latter removes the table Customers:

```
SELECT * FROM products;
DROP TABLE customers;
```

- This string, sent as a SQL statement, will lead to the execution of 2 statements. The products will be listed and, additionaly, the customers' table will be removed, all its data being lost as a consequence.

5. Databases' Vulnerability

- In the systems which do not allow stacked queries or which invalidate SQL strings containing ; these queries are not executed.
- SQL Injection can be prevented by the validation of the user's input.
- 3 approaches to the validation of the strings that represent queries are used: the use of a black list, of a white list or the implementation of the parametrized queries.

5. Databases’ Vulnerability

- **The black list** parses the input string, comparing each character to those in a list of not allowed characters. Disadvantage: many special characters can be legitimate, but will be rejected (example: the use of a quote character in the name).
- **The white list** is similar to the black one, but the comparison is made with the help of a list of allowed characters instead. The approach is more preferred than the previous one, but there are special considerations to be taken into account regarding the quote character.
- The parametrized queries use internally defined parameters to complete a previously prepared SQL statement.
- The **input validation** is one of the main mechanisms of defense for the prevention of database vulnerabilities, including SQL injection.

6. Inference in databases

- A subtle vulnerability in databases
- The ability to infer unknown information based on the already retrieved one
- The **problem**: there are no ideal solutions
- The only accepted solutions include queries' control (removal) or individual database items control (concealment)
- The requested sensitive data are either not provided, or the responses are close matches, but not exact.

6. Inference in databases

- An aspect of the inference appears when the initial goal is to generate or view aggregate values, provided that there were no privileges granted to obtain the individual values.
- Individual values can sometimes be inferred from the aggregate values

6. Inference in databases

- **Example**: An employee wants to find out the salary of his colleague X, which is confidential. The employee has rights to generate aggregate data (the average of the salaries based on certain criteria).
  - Suppose that X is a woman and she has 8 subordinates
  - Based on this information, the employee can use an aggregate function:

```
SELECT AVG(salary)
FROM employees
WHERE gender =‘F’ and subords = 8;
```

  - The salary of X can be obtained

6. Inference in databases

- Inference can also appear when the users can determine information from data that is accessible to them, at their security level, even if the information is protected at a higher level of security.
- **Example:** Some data (referring to the prototypes of the company's products) are not accessible to the junior employees. These employees have the right to update the table *Storage*, which records the content of the storage areas of the company, but they cannot read the rows which refer to the prototypes.
  - If the employee tries to modify a protected row, an error message will be raised
  - The message indicates the fact that this information is hidden and it can be deduced that a prototype is stored in the compartment referred in the update statement.
- Possible solution: **polyinstantiation**
- This allows the database to store more rows with "the same primary key"; they are distinguished by an identifier of the security level.

6. Inference in databases

- The development of technological solutions to detect the inference is a complex problem.
- Many papers in the field propose the withdrawal of the access to certain objects, based on the user's history.
- The problem with detecting the inference: it leads to a significant delay between the moment of the query execution and the moment of displaying the results.
- Other aproaches regarding the attenuation of the vulnerabilities emphasize the need of compromises.
- The protection of the sensitive data needs the inspection of situations which can lead to the exposure of data to unauthorized users, but also needs monitoring policies that must be implemented in order to ensure the retrieval of adequate responses.

7. Auditing in Database Security

- Database auditing is the process of monitoring and recording activities that occur within the database to ensure accountability and traceability of user actions.
- Main Objectives:
  - Detect unauthorized or suspicious activity
  - Support post-incident investigationVerify compliance with internal or legal requirements (e.g., GDPR, HIPAA)
  - Maintain accountability for all database users
- Core Principle: You cannot protect what you do not monitor.

7. Auditing in Database Security

- Commonly audited actions:
  - Logins and logouts
  - DDL operations (CREATE, ALTER, DROP)
  - DML operations (INSERT, UPDATE, DELETE, SELECT on sensitive tables)
  - Privilege escalations and role changes
  - Access to security-critical objects or system catalogs
- Audit Trail:Stored in dedicated log tables or files, containing:
  - Who performed the action
  - When it occurred
  - What was executed
  - From where (terminal, IP, application)
- Output Options: Database audit trail, OS audit file, or external security repository.

8. What’s new in Oracle 23ai? (1/2)

- 🧱 **SQL Firewall**: new feature that inspects and blocks unauthorized SQL statements, preventing SQL injection attacks.
- **TLS 1.3 Support:** Oracle 23ai now supports the Transport Layer Security 1.3 protocol, ensuring improved security for data in transit.
- **Increased Password Length:** The maximum password length has been extended to 1024 bytes, providing increased protection against brute-force attacks.
- **Multi-Factor Authentication (MFA):** It is now possible to enable multi-factor authentication for native database users.

8. What's new in Oracle 23ai? (2/2)

- **AI-Based Anomaly Detection:** Uses artificial intelligence to monitor data access and detect suspicious activities based on context (location, time, etc.).
- **Intelligent Data Masking:** Data masking is now dynamic and sensitive to user roles, displaying data differently depending on who is accessing it.
- **Advanced Auditing with AI Logs:** Audit logs are analyzed with AI to interpret user behavior and anticipate insider threats.
- **Self-Monitoring Encryption (TDE 2.0):** Encryption keys now have usage profiles and can trigger alerts in case of abnormal use.
- **Schema-Level Privileges and Read-Only Users:** Provides more granular control over privileges and introduces dedicated roles for read-only access.
- **Microsoft Entra ID (Azure AD) OAuth2 Integration:** Simplifies authentication for cloud users.

https://www.dbta.com/Editorial/Quest-IOUG-Database--Technology-Community-News/New-Database-Security-Enhancements-in-Oracle-23ai-168317.aspx
https://www.datapatroltech.com/blog/data-security-with-oracle-database-whats-new-in-2025