

Essay

Database attacks: threats and defenses

Databases are crucial to modern organizations, storing sensitive information such as financial information, personal data and business intelligence which makes them frequent targets for cyberattacks. If the attack turns out to be successful severe consequences can occur like financial loss, legal penalties... this essay examines common database attacks, their impacts, real-world examples and the possible defense mechanisms.

Types of database attacks

SQL injection

SQL Injection (SQLi) is one of the most predominant attacks: it exploits poorly sanitized input fields, allowing attackers to manipulate SQL queries. For example, entering

'OR '1'='1 in a login field can bypass the authentication. SQLi then enables unauthorized data access, modification, or deletion [1].

Privilege escalation

Privilege escalation occurs when attackers gain higher access rights than permitted. They may exploit vulnerabilities or misconfigured permissions to move from a basic user account to administrative control, granting them full database access [2].

Denial of service (DoS)

Attackers can overload a database with excessive queries, leading to Denial of Service (DoS). Although data may not be stolen, critical services will become unavailable, disrupting operations [3].

Insider threats

Employees or contractors with legitimate access can intentionally or not compromise databases. Insiders can leak data, abuse privileges, or sabotage systems, often with knowledge of sensitive operations [4].

Malware and Ransomware

Databases can also be targeted with malware. Ransomware attacks encrypt records and demand payment for decryption keys, a growing trend in enterprise environments [5].

Consequences of Database attacks

The effects of these attacks can go far beyond lost data.

- Financial loss caused by the ransomware, legal fines and response costs.
- Reputational harm reduces customer trust and loyalty
- Legal consequences can occur when organizations fail to conform with data protection laws such as GDPR or HIPAA.

For example, a well-known case is the 2017 Equifax breach, which led to the exposition of 147 million individuals' personal data leaked and resulted in a settlement of more than 700 million dollars [6].

Real world examples

Several well-known breaches highlight the risks of poor data protection.

- Yahoo (2013 – 2014): attackers stole data from three billion accounts, the largest known breach [7].
- Marriott International (2018): a breach that exposed details of 500 million customers, including their passport numbers [8].
- MongoDB Ransom attacks (2017): thousands of misconfigured databases were hijacked with attackers demanding payment [9].

These incidents demonstrate how both technical flaws and mismanagement can lead to catastrophic outcomes.

Defense mechanisms

Organizations can adopt layered defense mechanisms to protect their databases:

- Input validation and separated statements: Applications should validate input and use parameterized queries to block SQL Injection [10].
- Access control: Role-based access control (RBAC) and multi-factor authentication can ensure users only access necessary resources [11].
- Encryption: Encrypting data both at rest and in transit can reduce the risk of exposure if databases are compromised [12].
- Monitoring and Auditing: Regular logging, monitoring, and intrusion detection systems may help identify suspicious behavior before major damage occurs [13].
- Backups and disaster recovery: adopting secure routine backups and enabling recovery from ransomware or corruption can minimize damages [14].

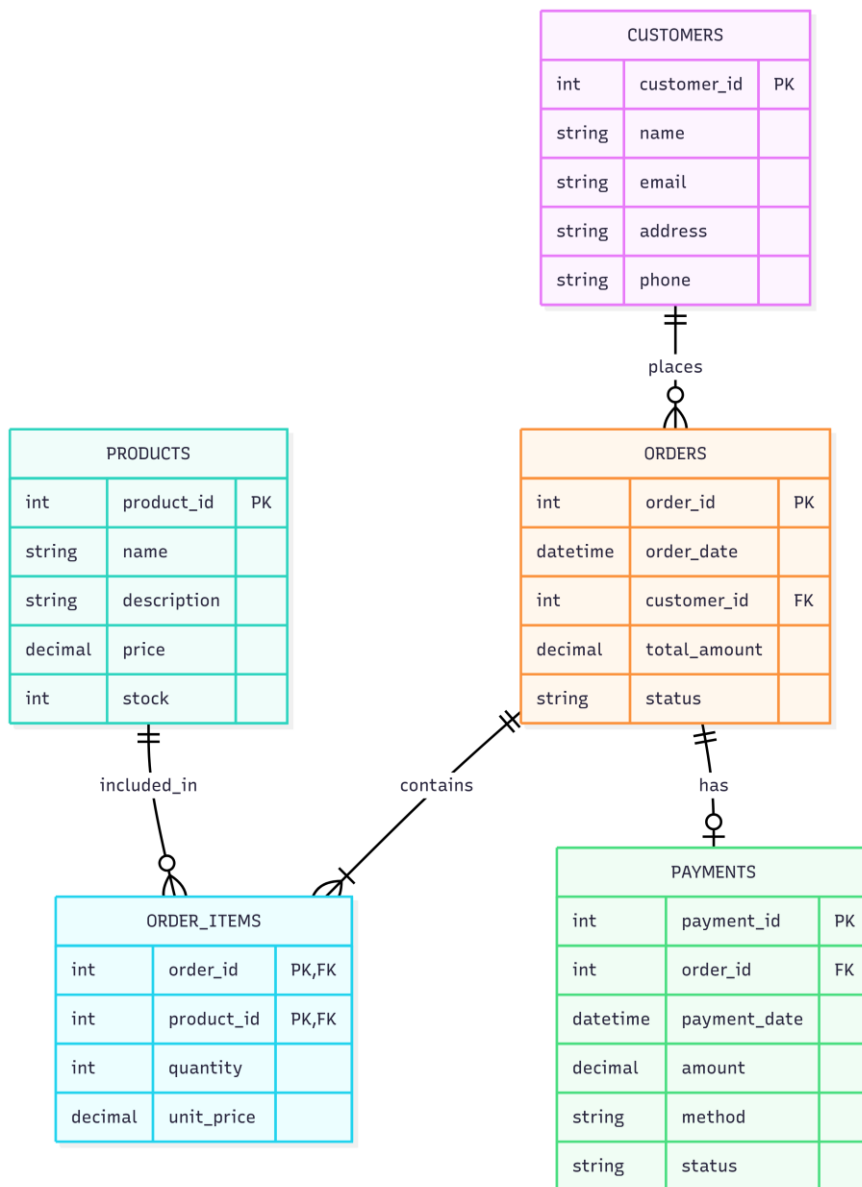
Conclusion

Database attacks threaten the confidentiality, integrity and availability of critical data. Techniques such as SQL Injection, privilege escalation, insider abuse and ransomware continue to endanger organizations on a worldwide scale. However, adopting security practices like input validation, strict access control, encryption, monitoring and reliable backup strategies can significantly reduce the risks. In a data-driven world, database protection is essential for organizational resilience and trust.

References (IEEE style)

- [1] M. Halfond, J. Viegas, and A. Orso, "A classification of SQL-injection attacks and countermeasures," *Proc. IEEE Intl. Symp. Secure Software Eng.*, 2006.
- [2] R. Ramakrishnan and J. Gehrke, *Database Management Systems*, 3rd ed., McGraw-Hill, 2003.
- [3] A. Bhuyan et al., "Distributed denial of service attacks: Types, detection, and mitigation techniques," *Int. J. Netw. Secur.*, vol. 17, no. 3, pp. 291–310, 2015.
- [4] M. Bishop and C. Gates, "Defining the insider threat," *Proc. IEEE S&P Workshops*, pp. 63–67, 2008.
- [5] Sophos, "The State of Ransomware 2020," [Online]. Available: <https://www.sophos.com/ransomware>.
- [6] U.S. Federal Trade Commission, "Equifax data breach settlement," 2019. [Online]. Available: <https://www.ftc.gov/equifax>.
- [7] Verizon, "2017 Data Breach Investigations Report," 2017.
- [8] BBC News, "Marriott data breach: 500m guests affected," Nov. 2018. [Online]. Available: <https://www.bbc.com/news>.
- [9] T. Hunt, "The MongoDB ransom attacks," 2017. [Online]. Available: <https://www.troyhunt.com>.
- [10] OWASP, "SQL Injection Prevention Cheat Sheet," 2021. [Online]. Available: <https://owasp.org>.
- [11] NIST, "Digital Identity Guidelines," *NIST SP 800-63-3*, 2017.
- [12] ISO, "ISO/IEC 27040:2015 Information technology — Security techniques — Storage security," 2015.
- [13] S. Axelsson, "Intrusion detection systems: A survey and taxonomy," *Chalmers Univ. Tech. Rep.*, 2000.
- [14] IBM, "Business continuity and disaster recovery," 2020. [Online]. Available: <https://www.ibm.com>.

Database Model Diagram



Explanation of Entities and Relationships

The online store database consists of five main entities: Customers, Products, Orders, Order_Items, and Payments. The relationships between these entities define how the store manages products, customers, and transactions.

Entities

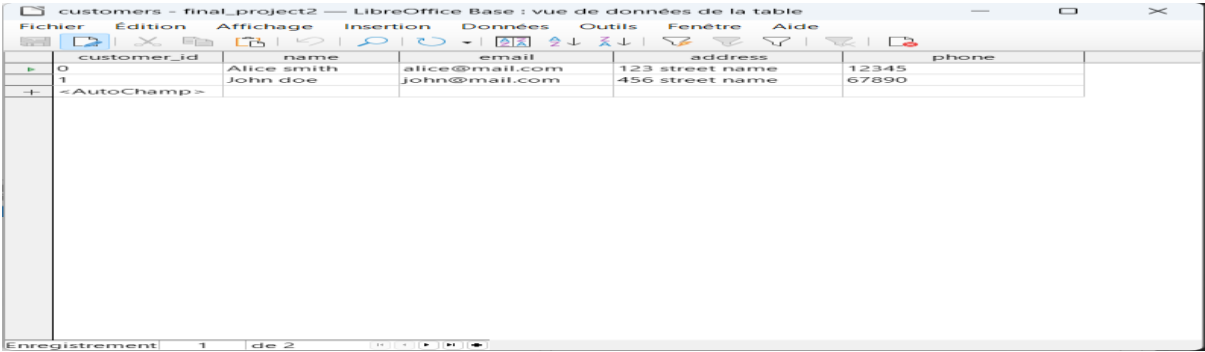
1. Customers

- Stores information about registered users.
 - Attributes: customer_id (PK), name, email, address, phone.
 - Each customer can place multiple orders.
2. Products
- Contains details of items available in the store.
 - Attributes: product_id (PK), name, description, price, stock.
 - Products can appear in multiple orders through the Order_Items table.
3. Orders
- Represents purchases made by customers.
 - Attributes: order_id (PK), order_date, customer_id (FK), total_amount, status.
 - Each order belongs to a single customer and can include multiple products.
4. Order_Items
- Junction table connecting Orders and Products, modeling a many-to-many relationship.
 - Attributes: order_id (FK), product_id (FK), quantity, unit_price.
 - Primary key is a composite of (order_id, product_id).
5. Payments
- Tracks payment details for each order.
 - Attributes: payment_id (PK), order_id (FK), payment_date, amount, method, status.
 - An order can have one or more payments, supporting partial or multiple payments.

Database implementation

Table structures

customers

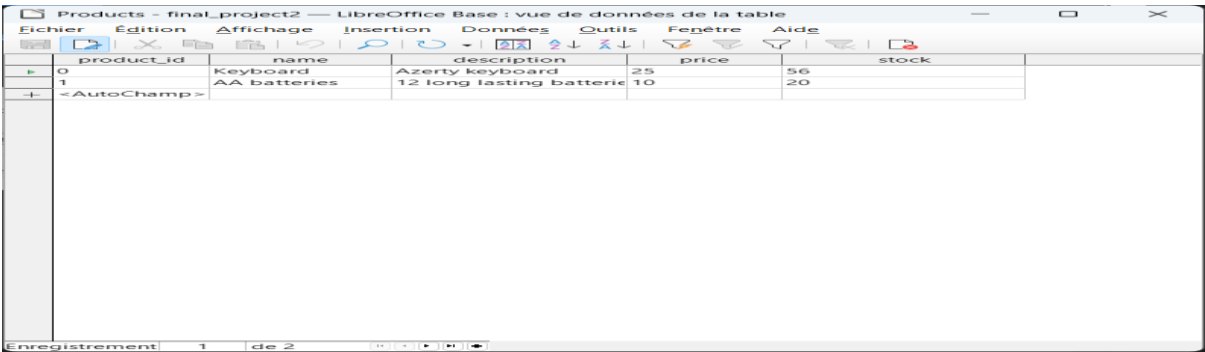


The screenshot shows the 'customers' table in LibreOffice Base. The table has six columns: customer_id, name, email, address, phone, and an empty column. The data is as follows:

customer_id	name	email	address	phone	
0	Alice smith	alice@mail.com	123 street name	12345	
1	John doe	john@mail.com	456 street name	67890	
<AutoChamp>					

The status bar at the bottom indicates 'Enregistrement 1 de 2'.

Products

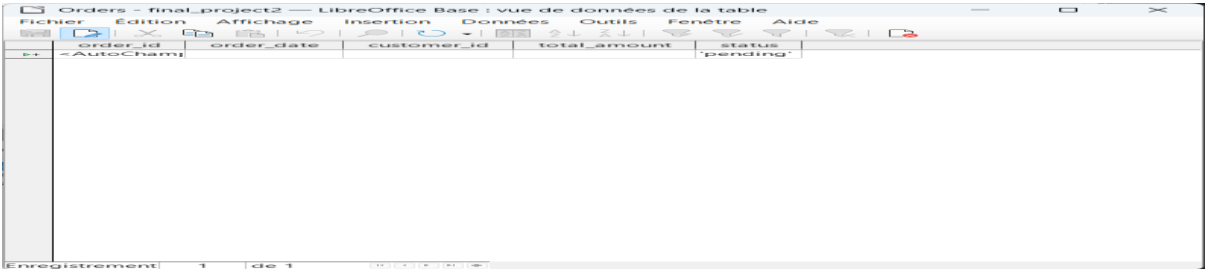


The screenshot shows the 'Products' table in LibreOffice Base. The table has six columns: product_id, name, description, price, stock, and an empty column. The data is as follows:

product_id	name	description	price	stock	
0	Keyboard	Azerty keyboard	25	56	
1	AA batteries	12 long lasting batterie	10	20	
<AutoChamp>					

The status bar at the bottom indicates 'Enregistrement 1 de 2'.

Orders

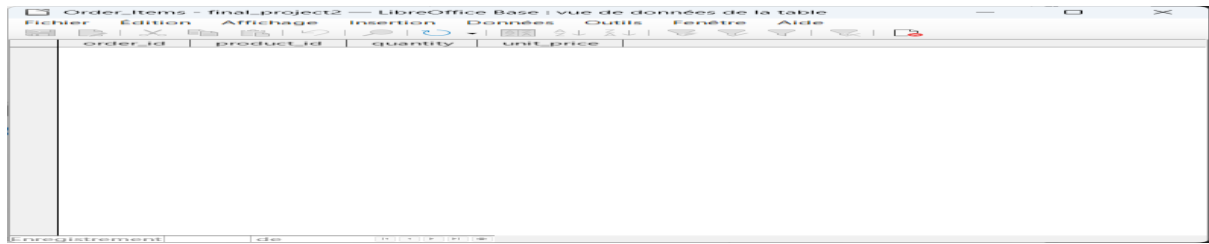


The screenshot shows the 'Orders' table in LibreOffice Base. The table has five columns: order_id, order_date, customer_id, total_amount, and status. The data is as follows:

order_id	order_date	customer_id	total_amount	status
<AutoChamp>				'pending'

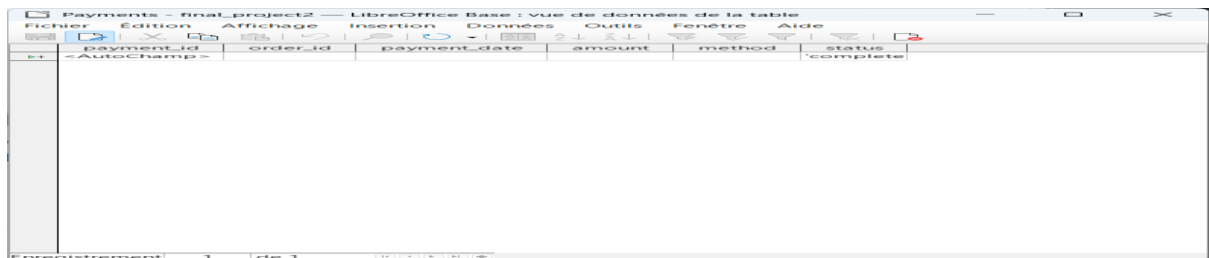
The status bar at the bottom indicates 'Enregistrement 1 de 1'.

Order Items



order_id	product_id	quantity	unit_price
----------	------------	----------	------------

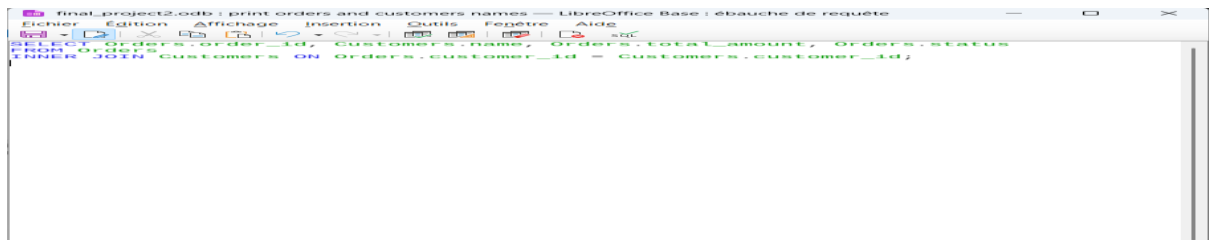
Payments



payment_id	order_id	payment_date	amount	method	status
------------	----------	--------------	--------	--------	--------

Queries

- Print all orders and customers names



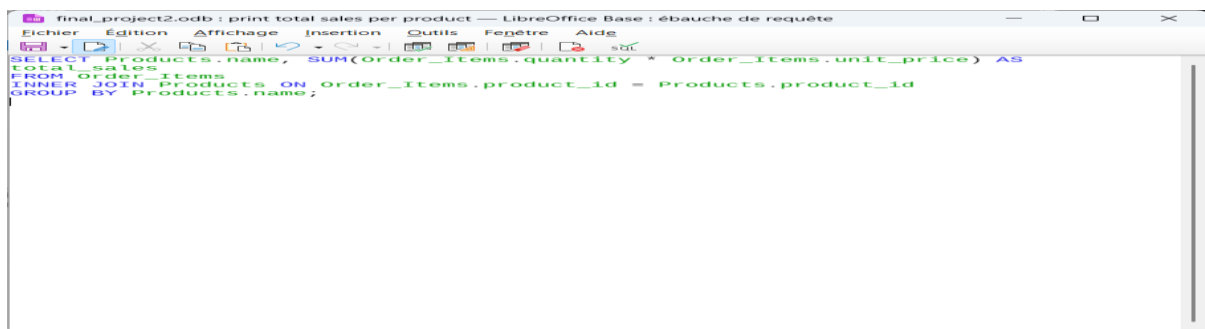
```
SELECT Orders.order_id, Customers.name, Orders.Total_Amount, Orders.Status
FROM Orders
INNER JOIN Customers ON Orders.customer_id = Customers.customer_id;
```

- Print products with stock <30



```
SELECT name, stock FROM Products WHERE stock < 30;
```

- Print total sales per products



The screenshot shows the LibreOffice Base query editor window titled "final_project2.odb : print total sales per product — LibreOffice Base : ébauche de requête". The menu bar includes "Fichier", "Édition", "Affichage", "Insertion", "Outils", "Fenêtre", and "Aide". The toolbar contains icons for opening, saving, and running queries. The SQL query entered is:

```
SELECT Products.name, SUM(Order_Items.quantity * Order_Items.unit_price) AS  
Total_Sales  
FROM Order_Items  
INNER JOIN Products ON Order_Items.product_id = Products.product_id  
GROUP BY Products.name;
```

- Print customers order history



The screenshot shows the LibreOffice Base query editor window titled "final_project2.odb : print customers order history — LibreOffice Base : ébauche de requête". The menu bar includes "Fichier", "Édition", "Affichage", "Insertion", "Outils", "Fenêtre", and "Aide". The toolbar contains icons for opening, saving, and running queries. The SQL query entered is:

```
SELECT Customers.name, Orders.order_id, Orders.order_date, Orders.total_amount  
FROM Orders  
INNER JOIN Customers ON Orders.customer_id = Customers.customer_id  
WHERE Customers.name = 'Alice Smith';
```