Lecture 10. Database Security (Chapter 30)

alisa.lincke@lnu.se

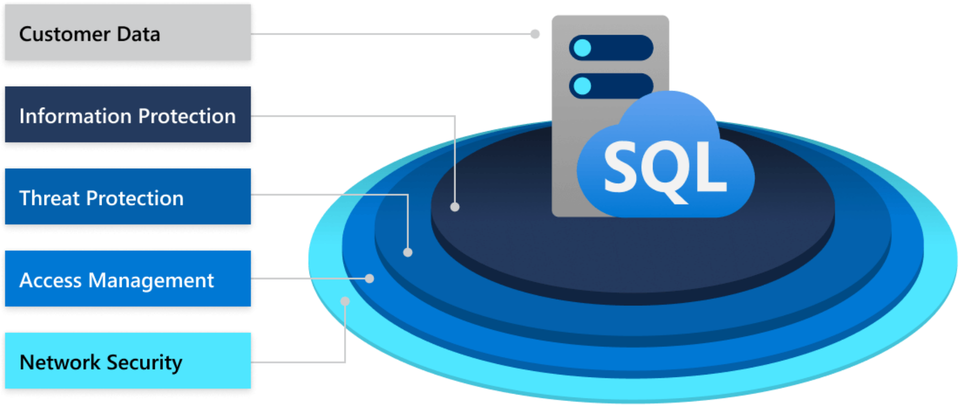
Linnæus University

Outline

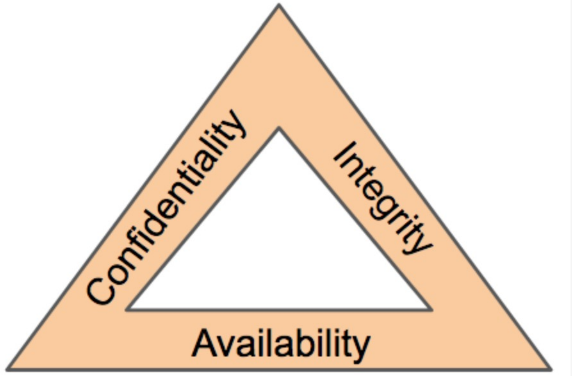
- Database Security and the DBA
- Sensitive Data and Types of Disclosures
- Privileges and Role-based Access Control
- Break 10 min
- SQL Injection and Protection Techniques against SQL Injection

Database Security

- The objective of database security is to secure sensitive data and maintain the confidentiality, integrity, and availability, (CIA) of the database [1].
- Database security protects the database management system and associated applications, systems, physical and virtual servers, and network infrastructure [1].



What is CIA Triad?



- Confidentiality
  - Protect information from unauthorized access and misuse
  - Protect sensitive information according to GDPR regulations
- Integrity
  - Improper modification of information or unauthorized alternation
  - Provide assurance in the accuracy and completeness of data
  - Secure access control on the system level (e.g., system users are only able to alter information that they are legitimately authorized to alter)
- Availability
  - Must be available to authorized users

Outsiders and Insiders

- Outsiders include anyone from lone hackers to cybercriminals seeking business disruption
- Insiders may comprise current or former employees, curiosity seekers, and customers or partners who take advantage of their position of trust to steal data, or who make a mistake resulting in an unintended security event
- Both outsiders and insiders create risk for the security of personal data, financial data, trade secrets, and regulated data

Threats of Security

- Insider Threats:
  - A malicious insider with bed intent
  - A negligent person within the organization who exposes the database to attack through careless actions
  - Human error: weak passwords, password sharing, configuration mistakes, and other irresponsible user behaviors which cause nearly 90% of security breaches
- Outsider Threats:
  - SQL/NoSQL injection Attacks
  - Compromising or stealing the credentials of a privileged administrator or application.
  - Stealing data from nonproductive environments such as DevTest which are usually not encrypted
- Database Management System Vulnerabilities

Database Security Layers [4]

Security Level	Description	Database Security Solutions
Physical level	The organization has own data center, servers, own cloud services. This level is vulnerable for infrastructure damage due to physical/natural disaster, human accidents, and malicious attacks from internal or external personnel.	Security of premise (locks, camera, security personnel, accessed by authorized individuals, access is recorded, logged). Security of data centers
Network Level	The data communication happens via network.	HTTPs protocols, VPN or SSH connection, block all public network access to database servers, firewalls
Operation System Level		Regular security updates, patches updates
Database level	Sensitive information stored separately, GDPR	Privileges and access control, Data encryption, backup encryption,
DBMS level		Control Access, Strong passwords, regular security updates (patches)
Application level	Decides authorized access to the backend. This level of security should ensure attacker should not get control on hardware and other applications	Authentication, web application firewall (WAF)

Database Security Level Threats [4]

Threat	Description	Suggestions
Data loss and leakage	Unauthorized updating, deletion, removal or extraction of data	• Data encryption at rest • Authentication and authorization • backup and retention policies • Secure APIs and Data integrity checks should be implemented
Access data and control	Due to lack of access control mechanism, confidential information can be seen or used by authorized users	Access control mechanism should be implemented Key based access, various encryption techniques

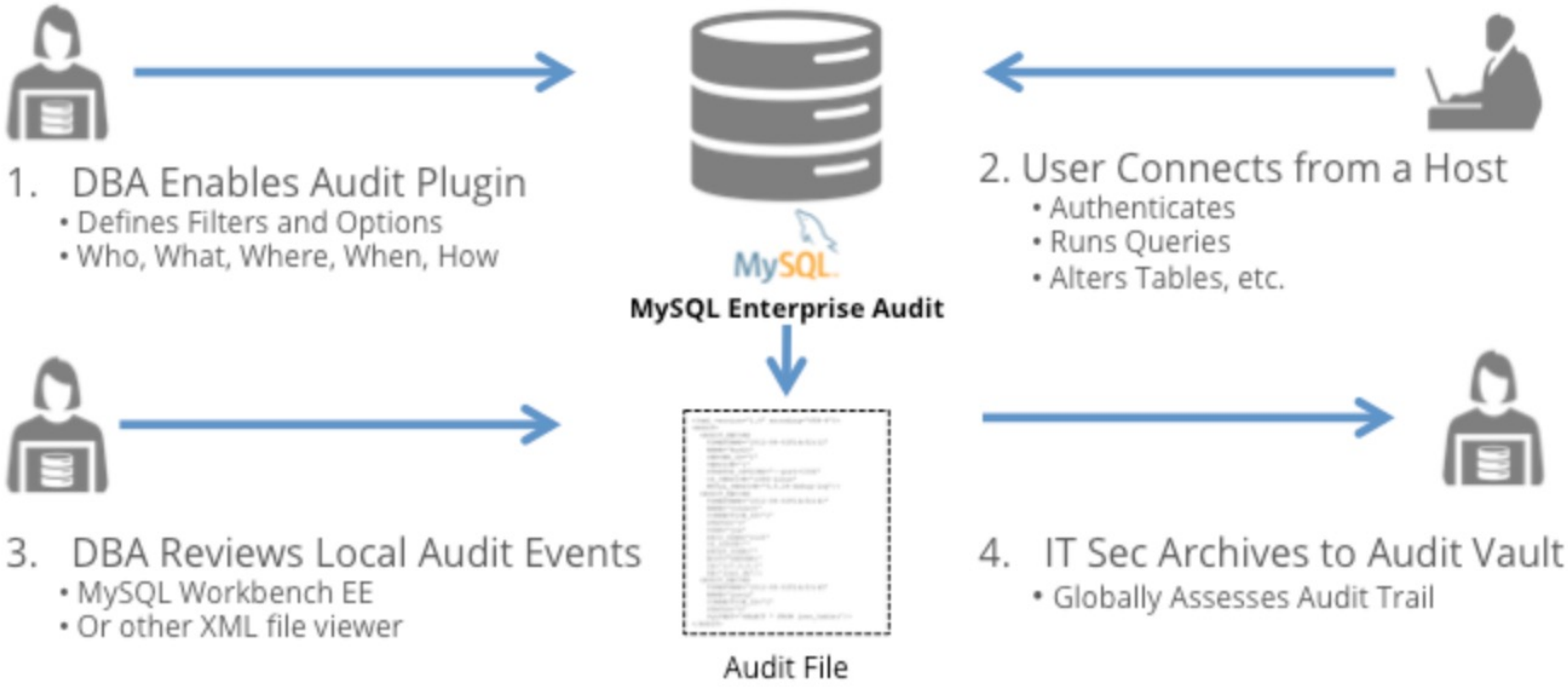
Database Administrator (DBA)

- Classifies users and data in accordance with the policy of the organization
- DBA has superuser account which provides powerful capabilities that are not available to regular database accounts and users.
- DBA is responsible for overall security of the database system.
- Has privileged commands to perform the following actions:
  - Account creation
  - Privilege granting
  - Privilege revocation
  - Security level assignment

Access Control, User Accounts, and Database Audits

- A person or group request an database account, and which data will be accessed
- DBA creates new account number and password with a certain privileges rights (remove ,create, alternate, read, write, etc.)
- The DBA usually creates table with all users which have access to the database, this table is encrypted with two columns: account number and password.
- The database system must also keep track of all operations on the database that are applied by a certain user (using System Log files).
- When the user is login, the DBMS can record the account number with associated device/computer. Is important to keep track of the database alternation operations (update, delete).
- Database audit is performed to find **illegal or unauthorized operations**
- **Access control** is done by granting and revoking of privileges

MySQL Enterprise Audit



1. DBA Enables Audit Plugin
   - Defines Filters and Options
   - Who, What, Where, When, How

2. User Connects from a Host
   - Authenticates
   - Runs Queries
   - Alters Tables, etc.

MySQL Enterprise Audit

3. DBA Reviews Local Audit Events
   - MySQL Workbench EE
   - Or other XML file viewer

Audit File

4. IT Sec Archives to Audit Vault
   - Globally Assesses Audit Trail

https://www.mysql.com/products/enterprise/audit.html

Sensitive Data and GDPR

- Sensitivity of data is a measure of the importance assigned to the data by its owner for the purpose of protection.
- Some databases contain no sensitive data, while other only sensitive data, or both sensitive and not sensitive data.
- According to GDPR, sensitive data is a personal data, and "'personal data' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person;" [3]
- DBA must ensure additional security for columns containing the sensitive (or personal) information (use private/public key encryption on both sides database and application)

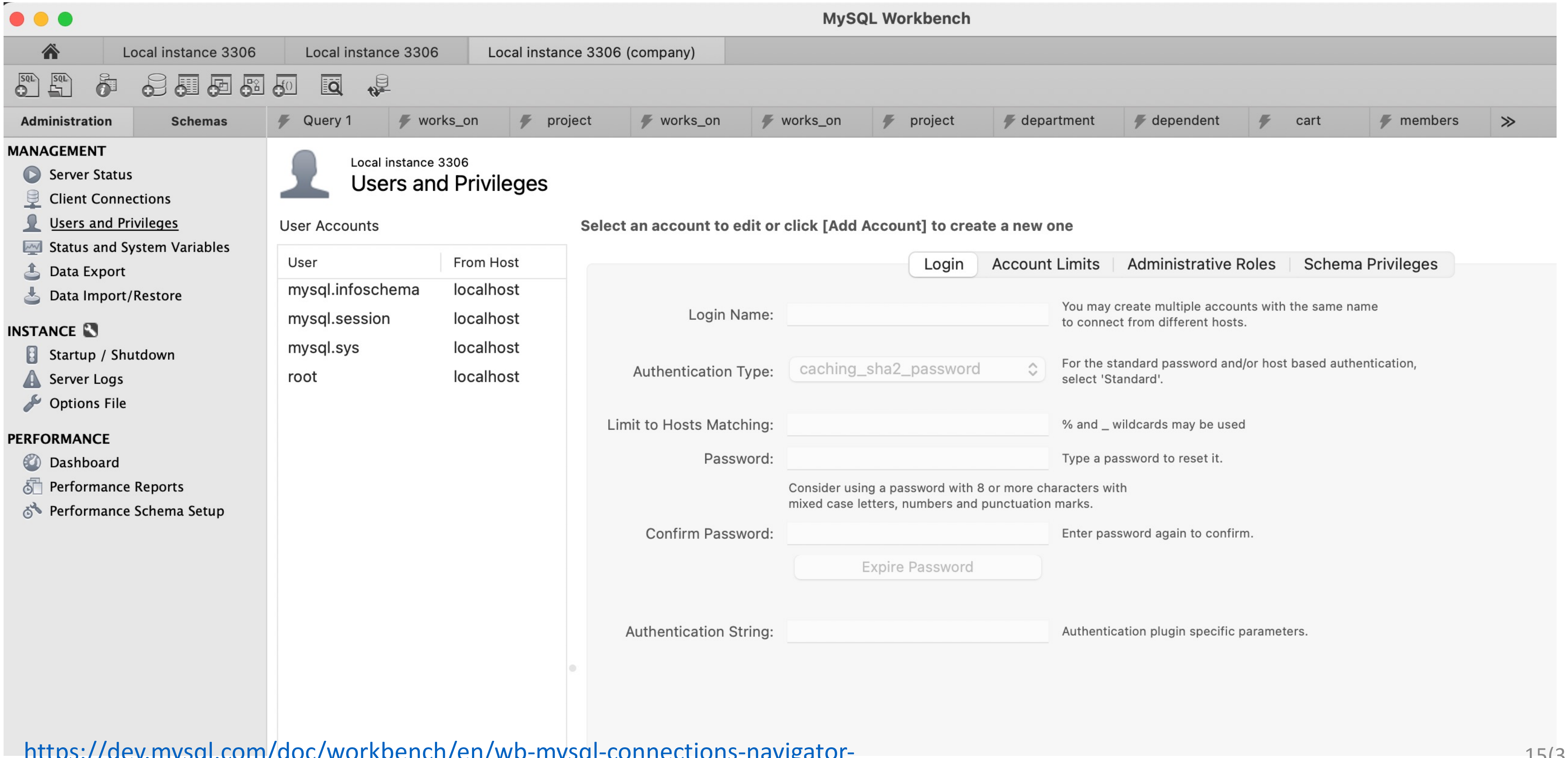
Privileges

- A privilege in a database management system is the permission to execute certain actions on the database.
- Two levels for assigning privileges:
  - Account level:
    - Example: CREATE, DROP, ALTER, SELECT privileges
  - Relation (or Table) level:
    - Example: DROP, DELETE, SELECT, UPDATE, INSERT privileges
- MySQL Privileges: https://dev.mysql.com/doc/mysql-security-excerpt/5.7/en/privileges-provided.html

Access Matrix Model

- The granting and revoking privileges organized so called the **access matrix model**, where**:**
  - Rows represents *subjects* (users, accounts, programs)
  - Columns represents *objects* (relations, records, columns, views, operations)
  - Each position in a matrix *M(i,j)* represent the type of privileges (*read, write, update*)
  - *read* (SELECT), *write* (INSERT), and *update* (DELETE, UPDATE, INSERT) privileges
- The one who created the SCHEMA is the owner account and has right to grand or revoke privileges.
- Granting and revoking can be done in two ways:
  - Using Views (Recommended)
  - Using Grant Option

MySQL Workbench Users and Privileges



Local instance 3306

Users and Privileges

User Accounts

User	From Host
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
root	localhost

Select an account to edit or click [Add Account] to create a new one

Login Name: You may create multiple accounts with the same name to connect from different hosts.

Authentication Type: caching_sha2_password For the standard password and/or host based authentication, select 'Standard'.

Limit to Hosts Matching: % and _ wildcards may be used

Password: Type a password to reset it.

Consider using a password with 8 or more characters with mixed case letters, numbers and punctuation marks.

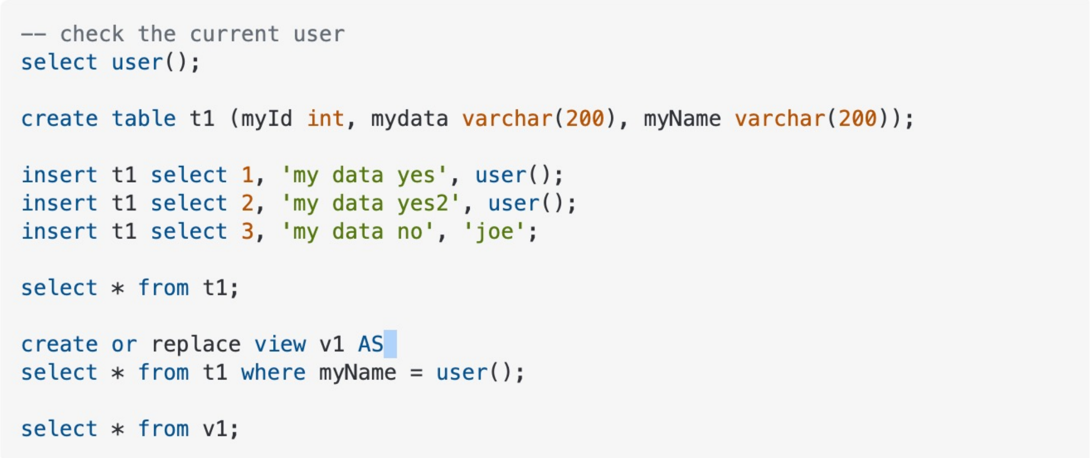
Confirm Password: Enter password again to confirm.

Authentication String: Authentication plugin specific parameters.

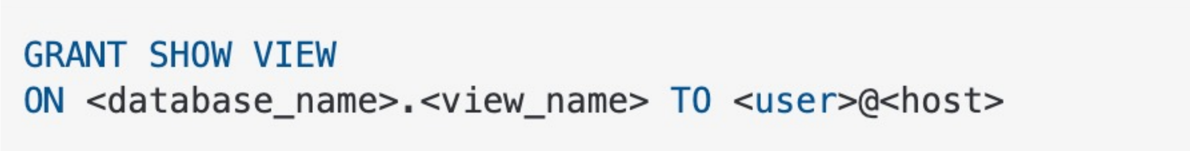
https://dev.mysql.com/doc/workbench/en/wb-mysql-connections-navigator-management-users-and-privileges.html

Use of Views

- Consider owner A of relation R and other party B
  - A can create view V of R that includes only attributes A wants B to access
    - Grant SELECT on V to B
- Can define the view with a query that selects only those tuples from R that A wants B to access



```
-- check the current user
select user();

create table t1 (myId int, mydata varchar(200), myName varchar(200));

insert t1 select 1, 'my data yes', user();
insert t1 select 2, 'my data yes2', user();
insert t1 select 3, 'my data no', 'joe';

select * from t1;

create or replace view v1 AS
select * from t1 where myName = user();

select * from v1;
```



```
GRANT SHOW VIEW
ON <database_name>.<view_name> TO <user>@<host>
```

Using the GRANT OPTION

- Propagation of privileges using the GRANT OPTION
  - If GRANT OPTION is given to B, the B can grant privilege to other account C, and so on. Solution: configure the limits on propagation
  - DBMS must keep track of how privileges were granted if DBMS allows propagation
- Revoking of Privileges
  - Useful for granting a privilege temporarily
  - REVOKE command used to cancel a privilege
- Syntax:

```
GRANT priv_type [, priv_type] ...
 ON object_type
TO user [user] ...
 [WITH GRANT OPTION ]
```

Examples: Granting/Revoking Privileges

- DBA to A1
    - GRANT CREATETAB TO A1;
    - CREATE SCHEMA Example AUTHORIZATION A1
    - A1 can create new tables
- A1 creates relations Employee and Department
- A1 to A2
    - GRANT INSERT DELETE on Employee, Department TO A2;
    - A2 was not given the WITH GRANT OPTION
    - A2 cannot give privilege to other users
- A1 to A3
    - GRANT SELECT On Employee, Department TO A3 WITH GRANT OPTION;
    - A3 given the WITH GRANT OPTION
    - A3 can give privilege to other users
- A3 to A4
    - GRANT SELECT On Emp TO A4;
    - A4 cannot propagate the SELECT privilege

Additional Security Level called Role-Based Access Control

- Introduced in 1994s to enforce security in large-scale enterprise-wide systems
- Permissions associated with organizational roles
  - Users are assigned to appropriate roles
- Mutual exclusion of roles
  - Both roles cannot be used simultaneously
  - Hence the role hierarchy are applied
- Identity management

Example:

```
GRANT ROLE full_time TO employee_type1
GRANT ROLE intern TO employee_type2
```

Other Additional Security Levels

- Label-Based Security:
    - Sophisticated access control rules implemented by considering the data row by row
    - Each row given a label
        - Used to prevent unauthorized users from viewing or altering certain data
    - Provides finer granularity of data security
    - Label security policy
        - Defined by an administrator
- Row-Level Access Control

Break 10 min

Web Application Security Level Threats [4]

Threat	Description	Suggestion
SQL injection attack	Attacker inserts a malicious code into SQL standard queries that gives him access to the database.	A strong user input detection and sanitization systems should be developed and implemented in the application
Cross-site scripting	Intruder adds a code/script into the web page which may be stored permanently or reflected just for the time on the web page	Various technologies like Web Application Vulnerability Detection Technology, Content Filtering, Content Based Data Leakage Prevention Technology etc. are available to detect and mitigate the attack
Cookie poisoning	Intruder can change the content of the cookie	Cookie saving should be disabled. Cookie cleanup is necessary
Backdoor and debug options	website debugging options if left by the developer then attacker can enter into the website easily and modify the content	At the time of website publishing, debug option should be disabled
Hidden field manipulation	Hidden fields are used by the developers to maintain the state. If it gets noticed then attacker can use to enter in the service	Use as less as possible of hidden fields and also query strings

SQL injection Attack (1)

- Attacker injects a string input through the application:
    - Changes or manipulates SQL statement to attacker’s advantage
- Types of attacks:
    - SQL Manipulation: changes the SQL command in the application, for example by adding conditions to the WHERE clause of a query
    - Or expanding a query components using set operations as UNION, INTERSECT etc.
    - Typical attack occurs during database login:
        - SELECT * FROM users WHERE username=“jake” and password=“jakespasswd”;
        - With SQL injection:
        - SELECT * FROM users WHERE username=“jake” and password=“jakespasswd” OR ‘x’=‘x’;

SQL injection Attack (2)

- Code Injection
  - The attacker can inject code into a program to change the course of execution
- Function Call Injection
  - a database function inserted into a vulnerable SQL statement to manipulate the data

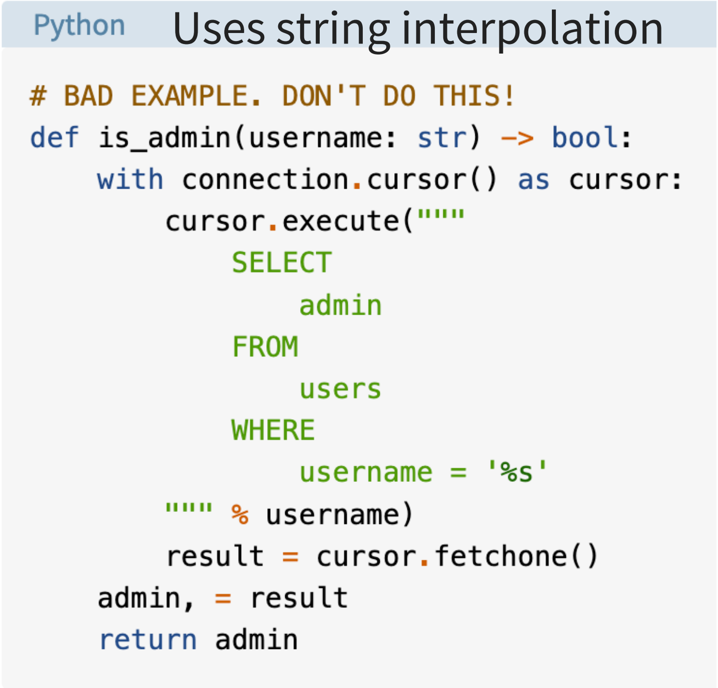
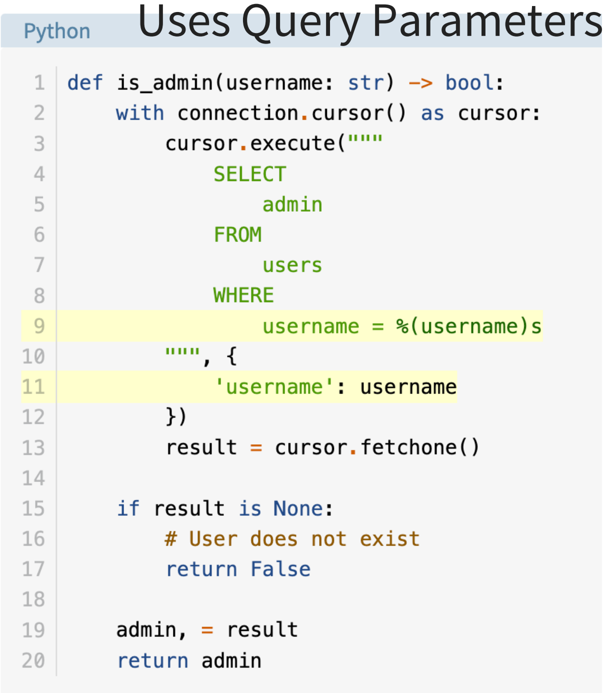
Risks Associated with SQL Injection

- Database fingerprinting (the type of database)
- Denial of service (flood the server)
- Bypassing authentication
- Identifying injectable parameters
- Executing remote commands
- Performing privilege escalation

Protection Techniques against SQL Injection

- **Bind Variables (Using Parameterized Statements)**

Example for Python Applications taken from https://realpython.com/prevent-python-sql-injection/

Not Secure Approach (good for SQL injection attacks)	Secure Approach to prevent SQL Injection attacks
	

Python Uses string interpolation

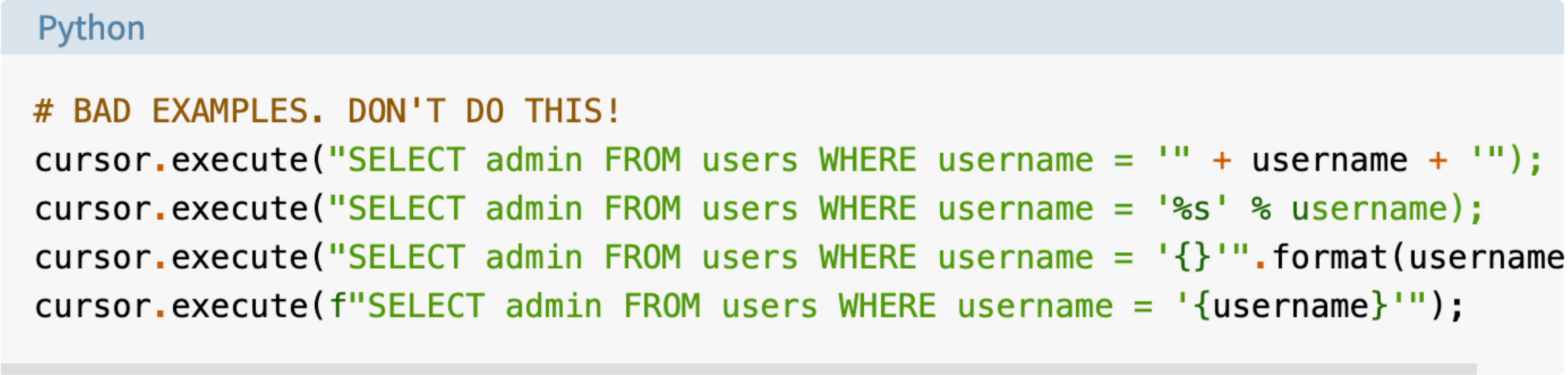
```python
# BAD EXAMPLE. DON'T DO THIS!
def is_admin(username: str) -> bool:
    with connection.cursor() as cursor:
        cursor.execute("""
            SELECT
                admin
            FROM
                users
            WHERE
                username = '%s'
        """ % username)
        result = cursor.fetchone()
    admin, = result
    return admin
```

Python Uses Query Parameters

```python
def is_admin(username: str) -> bool:
    with connection.cursor() as cursor:
        cursor.execute("""
            SELECT
                admin
            FROM
                users
            WHERE
                username = %(username)s
        """, {
            'username': username
        })
        result = cursor.fetchone()

    if result is None:
        # User does not exist
        return False

    admin, = result
    return admin
```

Bad SQL Query Examples

Not Secure



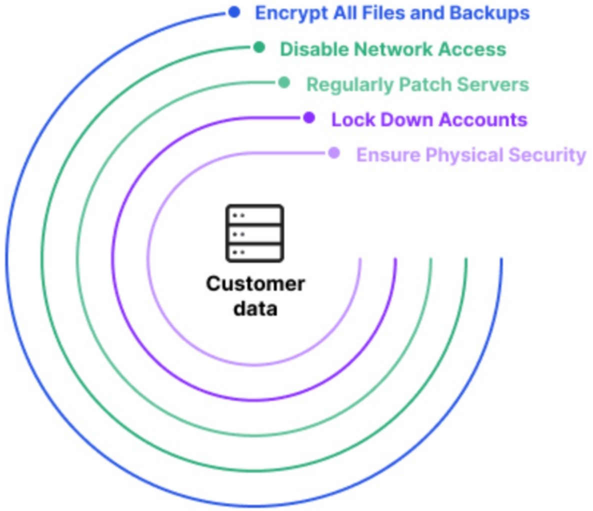
```python
# BAD EXAMPLES. DON'T DO THIS!
cursor.execute("SELECT admin FROM users WHERE username = '" + username + '");
cursor.execute("SELECT admin FROM users WHERE username = '%s' % username);
cursor.execute("SELECT admin FROM users WHERE username = '{}'".format(username
cursor.execute(f"SELECT admin FROM users WHERE username = '{username}'");
```

Secure SQL Query Examples

```
# SAFE EXAMPLES. DO THIS!
cursor.execute("SELECT admin FROM users WHERE username = %s'", (username, ));
cursor.execute("SELECT admin FROM users WHERE username = %(username)s", {'username':
username});
```

Database Security Best Practices

- Separate database servers from application server
- Isolate sensitive data from non-sensitive data
- Set up an HTTPS proxy server
- Avoid using default network ports
- Use real-time database monitoring
- Use database and web application firewalls
- Deploy data encryption protocols
- Create regular encrypted backups of your database
- Use strong user authentication
- Use security patches regularly in database management system
- Deploy regular vulnerability testing



How to store sensitive data?

- Use Secure Sockets Layer (SSL) is a standard security technology for establishing an encrypted link between a server and a client
- Use a secure encryption key
- The encrypted sensitive data can be stored as a BLOB type in MySQL and there are build in MySQL encryption functions
- Use encryption/decryption in the application code
- Transfer encrypted data over Internet
- Delete sensitive data which you no longer need
- Encrypt backups
- MySQL Enterprise TDE enables data-at-rest encryption by encrypting the physical files of the database. Data is encrypted automatically, in real time, prior to writing to storage and decrypted when read from storage. As a result, hackers and malicious users are unable to read sensitive data directly from database files.[5]

References

- [1] What is database security? Learn how to secure your database and protect it from threats: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-database-security/#what-is-database-security
- [2] Data Security and GDPR: https://www.oracle.com/se/security/database-security/what-is-data-security/
- [3] Art.4 GDPR Definitions: https://gdpr-info.eu/art-4-gdpr/
- [4] Kamatchi, R. & Ambekar, Kimaya & Parikh, Yash. (2017). Security Mapping of a Usage Based Cloud System. Network Protocols and Algorithms. 8. 56. 10.5296/npa.v8i4.10240.
- [5] MySQL Enterprise Transparent Data Encryption (TDE) https://www.mysql.com/products/enterprise/tde.html