1. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system.

2.Yes, Yes

3.Yes, select firstname from employees where lastname = ;

Let Lastname =” name’ OR ‘a’=’a “,

The addition of the OR 'a'='a' condition causes the where clause to always evaluate to true, so the query becomes logically equivalent to the much simpler query: select \* from employees

4. Let lastename equals to chenglong'); INSERT INTO employees(firstname, lastname) VALUES(‘chenglong’, ‘zheng’); --" . , so the query becomes logically equivalent to the much simpler query:

SELECT firstname FROM employees

WHERE lastname = 'chenglong’;

INSERT INTO employees(firstname, lastname) VALUES(‘chenglong’, ‘zheng’);

--'

5. Let lastname equals to ”chenglong’); UPDATE employee SET firstname=’chenglong’ WHERE lastname=’zheng’;-- “.so the query becomes logically equivalent to the much simpler query:

SELECT firstname FROM employees

WHERE lastname = 'chenglong’;

UPDATE employee  
SET firstname=’chenglong’  
WHERE lastname=’zheng’;

--‘

6.let lastname equals to "chenglong'); DELETE FROM employees where lastname = ‘deleteLastName’; --". so the query becomes logically equivalent to the much simpler query:

SELECT firstname FROM employees

WHERE lastname = 'chenglong’

DELETE FROM employees where lastname = ‘deleteLastName’;

--‘

7.

Primary Defenses:

* **Option #1: Use of Prepared Statements (Parameterized Queries)**
* **Option #2: Use of Stored Procedures**
* **Option #3: Escaping all User Supplied Input**

Additional Defenses:

* **Also Enforce: Least Privilege**
* **Also Perform: White List Input Validation**

Parameterized queries force the developer to first define all the SQL code, and then pass in each parameter to the query later. This coding style allows the database to distinguish between code and data, regardless of what user input is supplied.

Stored procedures have the same effect as the use of prepared statements when implemented safely\*. They require the developer to define the SQL code first, and then pass in the parameters after. The difference between prepared statements and stored procedures is that the SQL code for a stored procedure is defined and stored in the database itself, and then called from the application. Both of these techniques have the same effectiveness in preventing SQL injection so your organization should choose which approach makes the most sense for you.

This third technique is to escape user input before putting it in a query. If you are concerned that rewriting your dynamic queries as prepared statements or stored procedures might break your application or adversely affect performance, then this might be the best approach for you. However, this methodology is frail compared to using parameterized queries. This technique should only be used, with caution, to retrofit legacy code in a cost effective way. Applications built from scratch, or applications requiring low risk tolerance should be built or re-written using parameterized queries.

To minimize the potential damage of a successful SQL injection attack, you should minimize the privileges assigned to every database account in your environment. Start from the ground up to determine what access rights your application accounts require, rather than trying to figure out what access rights you need to take away. Make sure that accounts that only need read access are only granted read access to the tables they need access to. If an account only needs access to portions of a table, consider creating a view that limits access to that portion of the data and assigning the account access to the view instead, rather than the underlying table. Rarely, if ever, grant create or delete access to database accounts.

Input validation can be used to detect unauthorized input before it is passed to the SQL query