SQL Injection (SQLi) was discovered in 1998, but remains one of the top security concerns today. SQL injection a type of attack which takes advantage of vulnerabilities associated with the dynamic construction of SQL queries. In order to understand that, you need to recall that you can construct static SQL or dynamic SQL.

What is the difference between Static SQL and Dynamic SQL? Static SQL is SQL statements which are written in an application that do not change at runtime. You will hear these referred to as hard-coded or embedded SQL statements. These statement are preprocessed by a SQL preprocessor, which is database dependent, before the application is compiled. In the preprocessing stage, the database creates the access plan for each SQL statement, making the SQL static. Note that this is database dependent. Programs are written for access to specific databases.

After time and growth, there became a need to write code that was not database specific. In the early 1990’s, the first portable database API for SQL was defined. Following that came the ODBC specification from Microsoft. It was widely adopted and quickly became the de facto standard for SQL APIs. This meant that SQL did not have to be embedded into the application programming language, and precompilation was no longer required, which allowed for database independence. Thus the birth of dynamic SQL, which is SQL statements which are actually constructed at runtime. When you are using SQL APIs, the SQL in not embedded; it is dynamic.

This dynamic SQL leads to a software security breach which can allow attackers to insert code into a query. By doing so, they can use that to retrieve information that the attacker is not authorized to see, maliciously delete/modify data, or insert data that would give an attacker unauthorized access to the database. Some examples of what an attacker might wish to accomplish are:

* Control application behavior that’s based on data in the database, for example by tricking an application into allowing a login without a valid password
* Alter data in the database without authorization, for example by creating fraudulent records, adding users or “promoting” users to higher access levels, or deleting data
* Access data without authorization, for example by tricking the database into providing too many results for a query

As you might imagine, there are a lot of resources on the internet to help you understand SQL Injection. Here are a few to help you get started:

<https://www.w3schools.com/sql/sql_injection.asp>

<https://www.acunetix.com/websitesecurity/sql-injection/>

<http://www.unixwiz.net/techtips/sql-injection.html>

<https://www.owasp.org/index.php/SQL_injection>

<https://en.wikipedia.org/wiki/SQL_injection>

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>

<http://cwe.mitre.org/data/definitions/89.html>

Please research and thoughtfully answer the following questions:

Question 1 (8.7 from book): SQL Injection is often used to exploit a database. It refers to the process of using SQL injection to read sensitive data, modify a database, or execute administrative operations on a database. In preparation for an SQL infection exploitation, attackers often use SQL infection to discover information about the database. Investigate how SQL injection can be used to discover information about a database. In particular, how can SQL injection be used to discover information such as field name, tables name, or even email addresses?

SQL injection can be used in the WHERE clause of a query for example, adding an OR ‘a’ = ‘a’ which will always evaluate to true even if the other conditions are false. This will give the user access to unauthorized information in the query. In addition to this, some databases allow multiple SQL statements separated by semicolons in a single query string, which could cause even more problems with the potential of changes or loss of data.

Question 2: What is a parameterized query? How does that help avoid SQL injection attacks? Would this be helpful in a Stored Procedure and why?

A parameterized query is a prepared statement of the JDBC API which forces the values of variables that are used to construct a query to conform to a specific type value instead of an arbitrary string that can contain malicious SQL statements. This helps avoid SQL injection attacks because the queries are not dynamically constructed. This would be very helpful in a stored procedure because it would not give a user access to manipulate the query in a malicious way to gain unauthorized information.