SQL Injection or Insertion is still one of the top vulnerabilities according to OWASP Top10-2013. SQL Injection is consistently growing day by day and hence has become a buzz word. But what made this happen? It is due to the lack of security awareness during development of web applications. Developers are mandated to deliver functionality on time and on budget but not to develop secure applications, which results in development of vulnerable web applications. Inaccurate Security Requirements, Poor Design, Configuration Mistakes, Insecure or bad Coding Techniques, Complexity, Invalidated User Input, Password Management Flaws are major causes which make SQL Injection possible. To prevent or mitigate the risk of SQL Injection, there is a need to integrate security during development of web application.

What is SQL Injection?
“SQL Injection” refers to an attack where malicious users can inject SQL commands into an SQL statement, via input fields of web forms. Injected SQL commands can alter SQL statement and have an unauthorized access to the Database, there by compromising the security of a web application. SQL Injection attack mainly happens due to Input validation vulnerabilities. The common vulnerabilities that make SQL Injection possible in web application are:

- No or Improper user input validations.
- Constructing dynamic SQL queries using simple string concatenation.
- Configuring an application with an over privileged database login.
- Improper exception and error handling.
- To stop SQL Injection these vulnerabilities should be removed during development of web application.

SQL Injection Anatomy
Figure 1 illustrates an example for Tautology based attack, showing how SQL Injection happens. The attacker is attempting to put SQL commands to extract data from database. As shown in Fig. 1, attacker enters a tautology statement in textbox which is concatenated with the SQL Query at the backend and executed by the database.

What SQL Injection Can Do?
SQL Injection is a kind of attack which is very difficult to stop, because it happens as a normal functioning of web application. SQL Injection even bypasses the Authentication and Authorization of Web Application. It crosses the Network level (Firewalls and Intrusion detection System) and Operating system security. Web Application having SQL Injection Vulnerabilities is exposed to all types of threats as explained by STRIDE. STRIDE is a threat categorization model introduced by Microsoft. The acronym STRIDE (spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privilege) for threat categorization of SQL Injection is explained as below-

- S-Bypass Authentication and Authorization
- T-Steal and Modify Data
- R-Void or Delete Transactions and Drop Tables
- I-Disclosure of Sensitive Data
- D-Destroy Data and make it Unavailable
- E-Get and use Administrator credentials

It shows that SQL Injection is powerful attack which destroys the database and steals the billions of money from banks which leads to crisis in Organization. So, there is a need to stop SQL Injection before they stop you.

SQL Injection Mitigation
It is clear that, SQL Injection vulnerability is due to a flaw in Web Application Development. It is not a Database or Web Server Problem. When we talk about security, there are three ways to secure a web application -

1. Penetrate and Patch
2. Operational Environment
3. Secure Software Engineering

The dominating idea i.e. secure software engineering means addressing security during development, offers reduction in future expenditures, time as well as more in-depth defensive layers. So there is a need to take a holistic approach. Security should be weave in throughout the complete software development lifecycle starting from Requirements Phase to Testing phase. While designing a secure Web Application which is free from SQL Injection vulnerability, the three thumb rules, which should keep in mind are -

1. All Input is Evil
2. Defense In Depth
3. Think from Attackers Perspective

To mitigate SQL Injection, Different activities which should be performed during development are shown in Fig. 2. As applications are rarely static and need to be enhanced and adapted to suit changing business requirements, so vulnerabilities should be removed in their respective phase as the application evolves. Some measures to be taken in each phase to mitigate SQL Injection are explained below phase by phase -

Requirement Phase
Developing secure web applications that can withstand malicious SQL Injection
attacks requires a careful injection of security considerations into early stages of development lifecycle. Decisions taken in this phase will help us in implementing security in Design and coding phase. Following are some tasks which are necessary to perform for SQL Injection mitigation:-

- Incorporate Security Modelling (Misuse Case, Attack trees, Vulnerability Cause Graph) - Security modelling is a collective term for modelling techniques of security concepts such as threats, attacks and vulnerabilities. The security modelling identifies potential vulnerabilities, threats and countermeasures. Security Modelling is done to reduce the knowledge gap between Developers and Security Experts. The modelling techniques like vulnerability cause graphs (VCG), Attack trees and Misuse cases are used to elicit Security requirements for the mitigation of SQL Injection Vulnerability. The main role of these modelling techniques are-VCG- shows the causes of vulnerability in graphical form

- Attack Tree- shows how the system is threatened and exploited by attackers

- Misuse Case- is “Inverse Use Case” which shows the threats a vulnerability is exposed to and countermeasures to mitigate vulnerability.

- Elicit Accurate Security Requirements by using Security Modelling if Security modelling is properly done, then it is very easy for designers and developers to implement security to mitigate SQL Injection.

**Design Phase**
Now the system needs to be designed in such a way that all the security considerations have been taken into account. At Design time SQL Injection is prevented by –

- Proper Design review or audit
- Incorporate Threat Modeling.
- Data Flow diagram (DFD) and Architecture diagram Analysis
- Examine Entry and Exit points

**Coding phase**
During Coding, lot of vulnerabilities are introduced due to less skilled or unaware developers. SQL Injection vulnerability is introduced due to bad coding practices. To prevent SQL Injection following coding techniques should help:-

- Validate the user input (whitelisting/blacklisting)
- Never use Dynamic SQL queries by string concatenations.
- Use Parameterized commands with dynamic SQL queries.
- Stored Procedure is the best option to prevent attack.
- Implement the principle of Least privilege.
- All sensitive and confidential information like passwords should be stored in encrypted form.
- Implement strong client side as well as server side validation for all user inputs
- Use Regular expression to validate and limit the input data.
- Implement error handling, don’t show error messages to the user.
- Use Quotefunction
- Keeping untrusted data separate from backend commands and queries.
- Escape or filter or sanitize the special characters in user inputs.
- Use Exception handling to catch all possible exceptions.
- Set length limits, range on input data in form fields and validate data for content length and format.
- Make schema, table names unique.
- Try to avoid query strings for building Web pages.
- Audit the code to find vulnerabilities.

**Testing phase**
Security testing focuses on the testing potential security bugs that might be exploited by the hackers. Security testing goal is to ensure that the software being tested is robust and continues to function in an acceptable way even in existence of malicious attacks. During testing SQL Injection is mitigated as:-

- Ethical Hacking
- Perform penetration tests.
- Implement static and dynamic testing for code walkthroughs and inspection.
- Perform Fuzz Testing (Provide random unexpected inputs in input fields which are connected to a database and observe the outputs and error messages generated for the wrong inputs).
- Perform static code analysis or reviews

**GreenSQL**
GreenSQL is unified software solution provides Database Security, Dynamic Data Masking, and Database Activity Monitoring in one product. GreenSQL is an open source software as a proxy server (communication interface) or database firewall between the database server and web server is implemented. It includes a graphical user interface for configuring and monitoring a firewall. This supports Microsoft SQL Azure , SQL Server (all versions), MySQL . The software automatically checks queries on security and forwards them only after review. The logic is based on evaluation of SQL commands using a risk scoring matrix as well as blocking known database administrative commands (DROP, CREATE, etc). How it helps us –

- Blocks SQL Injection attacks
- Secures data
- Prevents Unauthorized Database Access
- Masks Sensitive Data

**Conclusions**
SQL Injection vulnerabilities are known for more than a decade, and they are still one of the most prevalent vulnerabilities in web applications. Today, wide variety of automated detection tools are available in the market which made easy to detect and exploit SQL injection (SQLI) vulnerabilities. SQLI vulnerabilities have high damage potential and can completely compromise the web application. So by raising the awareness and following few simple best practices during development of web applications will help completely preventing SQLI vulnerabilities.

**References**