Here is a **comprehensive, fully formatted, sentence-by-sentence breakdown** of the document **“SQL Injection Notes.docx”**, structured to match your requested layout and optimized for **CompTIA A+ 220-1102 exam prep** — particularly under:

* **Domain 2.0: Security**
* **Objective 2.5: Compare and contrast common threats, vulnerabilities, and attacks**

**🛡️ SQL Injection – Study Notes (CompTIA A+ 220-1102 Aligned)**

**✅ 1. Concept Overview: What Is an SQL Injection?**

**SQL (Structured Query Language)** is the language web applications use to communicate with database servers.

An **SQL injection** is a type of **code injection attack** where malicious SQL code is **inserted into input fields** to manipulate the back-end database.

* SQL injection targets applications that **interact with databases using user input**.
* It allows attackers to **bypass authentication**, **read or alter data**, and sometimes **take control** of the database server.

🧠 SQL injection = inserting SQL code into a web input field to **trick the system into running unintended database commands**.

**📘 2. Exam Relevance**

* **CompTIA A+ 220-1102 (Core 2)**
* **Domain 2.0: Security**
* **Objective 2.5:** Includes **SQL injection** as a specific threat type.

You’re expected to:

* Recognize SQL injection examples
* Understand how they exploit input fields
* Know how to prevent them using **input validation** and **least privilege**

**📋 3. Sentence-by-Sentence Breakdown**

**🧾 What Is SQL?**

* SQL = **Structured Query Language**
* Used by web applications to **send and retrieve data** from databases.
* Databases store **valuable information**, making SQL a **popular attack target**.

**🦠 What Is an SQL Injection?**

* SQL injection is the **insertion of malicious SQL code** into an input form (like a login box).
* It’s a specific type of **code injection attack**.
* Other common code injection types include:
  + **HTML injection**
  + **XML injection**
  + **LDAP injection**
* **SQL is the most commonly used attack** in this category.

**💻 How Does a Normal SQL Query Work?**

**Example Scenario:**

* A user tries to log in with:
  + Username: Jason
  + Password: pass123
* The web application sends this as a SQL query to the database:

SELECT \* FROM users WHERE username = 'Jason' AND password = 'pass123';

* If the data **matches a record**, access is granted.
* If the data **doesn’t match**, access is denied with an error.

**💣 How Does an SQL Injection Work?**

* An attacker inputs:
  + Username: Jason
  + Password:  `OR 1=1;
* The SQL query becomes:

SELECT \* FROM users WHERE username = 'Jason' AND password = '`OR 1=1;';

* The escape character (') **closes the intended input**, and the OR 1=1 part **always evaluates to TRUE**.
* This tricks the database into authenticating the attacker **without a valid password**.

**🚨 Why Is This Dangerous?**

* The database accepts 1=1 as **true**, which **bypasses normal security**.
* Attackers gain **unauthorized access** without needing real credentials.
* This allows access to sensitive or administrative areas of the application.

**🛡️ How to Prevent SQL Injections**

1. **Input Validation**
   * Check all user inputs before sending to the database.
   * Block or sanitize dangerous characters (e.g., ', ;, --, etc.).
2. **Least Privilege**
   * The web app should connect to the database with a **limited user account**, not an admin one.
   * Prevents attackers from doing full-scale damage even if they exploit a vulnerability.

🧠 Developers should **never trust user input** and must **validate everything**.

**🧠 Recognizing SQL Injections on the Exam**

* Look for questions containing:
  + OR 1=1
  + 7=7, 123=123, or any **always-true expressions**
* These are **classic SQL injection signs**.
* The correct answer will almost always involve:
  + **Input validation**
  + **Sanitizing user input**

**✅ Reminder About SQL’s Purpose**

* SQL itself is **not bad** — it’s essential for working with databases.
* The problem is **unsafe input** that gets turned into **dangerous SQL statements**.
* Protect the database by **controlling what commands get executed**.

**🧩 4. Key Table: Summary of Concepts**

| **Concept** | **Explanation** |
| --- | --- |
| **SQL Injection** | Inserting SQL code into an input field to manipulate database behavior |
| **Normal Login Query** | Authenticates users by comparing username/password to stored values |
| **Injection Example** | Using ' OR 1=1; to bypass authentication logic |
| **Result** | Unauthorized access to database or user session |
| **Defense 1** | Input validation (filter/escape dangerous input) |
| **Defense 2** | Least privilege (use minimal database permissions) |
| **Exam Keywords** | OR 1=1, 7=7, always-true, input validation, code injection |

**🧾 5. Exam Inclusion Notification**

✅ **Yes — SQL injection is directly covered in the CompTIA A+ 220-1102 exam**, under:

* **Objective 2.5**: Threats, vulnerabilities, and attacks
* You may be tested on:
  + Recognizing SQL injection syntax
  + Knowing prevention techniques
  + Differentiating SQL injection from other code injections

Would you like me to now generate a **10-question quiz** based on this breakdown to test your understanding?

Or would you prefer a **side-by-side comparison** of SQL vs LDAP vs XML injection types next?

Certainly! Below is a **professionally formatted, exam-style multiple-choice quiz** based on your **SQL Injection Notes**, designed specifically for the **CompTIA A+ 220-1102 (Core 2)** exam — **Objective 2.5: Threats, vulnerabilities, and attacks**.

This version is **formatted for Word**, with minimal spacing and clean alignment, so you can **copy and paste directly without reformatting**.