CSIS 330 – Lab 6: Identifying Security Vulnerabilities Answer Template

**Questions:**

1. **Answer the following questions in one or two sentences each: (1 point each)**
2. What is the Weakness Prevalence of this issue?
3. Describe how attackers can exploit this vulnerability.
4. List one “Architecture and Design” Prevention and Mitigation strategy that can be used to defend against this threat.
5. What is one “Implementation” approach used to defend against this threat?
6. List 2 CWEs of your choice that are related to this issue.
7. Highlight five of the ***Potential Mitigations*** Phases in the link provided and write a brief explanation of each.

**4. a.** This weakness typically appears in data-rich applications that save user inputs in a database. Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert additional statements that modify the back-end database, possibly including execution of system commands.

SQL injection has become a common issue with database-driven web sites. The flaw is easily detected, and easily exploited, and as such, any site or software package with even a minimal user base is likely to be subject to an attempted attack of this kind. This flaw depends on the fact that SQL makes no real distinction between the control and data planes.

**b.** If poor SQL commands are used to check usernames and passwords, it may be possible to connect to a system as another user with no previous knowledge of the password. Hackers enter partial SQL statements into user input fields that change the application’s underlying SQL and allow hackers to access information without the need for a password

**c.** Strategy: Enforcement by Conversion

When the set of acceptable objects, such as filenames or URLs, is limited or known, create a mapping from a set of fixed input values (such as numeric IDs) to the actual filenames or URLs, and reject all other inputs.

**d.** Ensure that error messages only contain minimal details that are useful to the intended audience and no one else. The messages need to strike the balance between being too cryptic (which can confuse users) or being too detailed (which may reveal more than intended). The messages should not reveal the methods that were used to determine the error. Attackers can use detailed information to refine or optimize their original attack, thereby increasing their chances of success.

If errors must be captured in some detail, record them in log messages, but consider what could occur if the log messages can be viewed by attackers. Highly sensitive information such as passwords should never be saved to log files.

Avoid inconsistent messaging that might accidentally tip off an attacker about internal state, such as whether a user account exists or not.

In the context of SQL Injection, error messages revealing the structure of a SQL query can help attackers tailor successful attack strings.

**e.** CWE-456: Missing Initialization of a Variable (<https://cwe.mitre.org/data/definitions/456.html>)

CWE-457: Use of Uninitialized Variable (<https://cwe.mitre.org/data/definitions/457.html>)

**5. a.** Libraries or Frameworks – Use a library that is designed to prevent against SQL injection, such as Dropwizard or Hibernate, which serve as built in persistence layers to protect against SQL injection

**b.** Parameterization – Develop structured code that separates data from code. When possible, avoid dynamically creating and executing query strings using “exec” mode, where hackers can access more information.

**c.** Environment Hardening – Run code only using the lowest privileges required. When creating user account for a SQL database, follow the principle of least privilege. Do not allow unpermitted users to read/update other user’s data.

**d.** Input Validation – perform a validation check on all inputted data to determine its validity for a given request. Create a strict validation that ensure no possible partial SQL statements can be inputted.

**e.** Enforcement by Conversion – create a mapping from fixed inputs to their actual names, URLs, files, etc. and only use the keys as SQL parameters. This creates a check for valid values and only uses currently stored key/value pairs. This will prevent hackers from directly injecting SQL statements into the code.

**Deliverables:**

Save your Answer Template using the convention of [your first initial] + [your last name] + “\_Lab6”.

For example: Joe Smith will save his file template as JSmith\_Lab6.doc .

Submit your **Word Template** to Blackboard by attaching it to the appropriate assignment link.