**Network Structure and Cloud Computing Assignment-10 Attack Report**

**Test Case 1:**

Name:

SQL Injection

Description:

SQL Injectionis a code injection technique, used to attack data-driven applications, in which nefarious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker). SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape character embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database

Example:

https://csye6225-fall2018-sanghavida.me//transaction?id=8a8080cd67512ea401675137df450000

Result:

1. Output is manipulated
2. Data did not return

Solution:

1. Never Trust Client-side Validation, Better to check it on server-side as well
2. Use of prepared statements
3. Do not generate dynamic queries using string concatenation
4. Grant minimum database access whenever necessary for application
5. Type check data on server-side.

Reference:

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

**Test Case 2:**

Name:

Application Error Disclosure

Description:

An important aspect of secure application development is to prevent information leakage. Error messages give an attacker great insight into the inner workings of an application. The purpose of reviewing the Error Handling code is to assure the application fails safely under all possible error conditions, expected and unexpected. No sensitive information is presented to the user when an error occurs

Example:

<https://csye6225-fall2018-sanghavida.me//transaction/8a8080cd67512ea401675137df45000/attachments>

Result:

1. Could leak sensitive information to attacker
2. Hard to debug for users on front-end
3. Can lead to exploitation of information

Solution:

1. Have a localized message for every generic error. Not exposing sensitive information in messages
2. Usage of try catch block in java based application
3. System exceptions for vulnerable code in C#
4. Releasing resource and good housekeeping.
5. Having a centralized Exception handling and Logging

Reference:

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

**Test Case 3:**

Name:

Unlimited size of request body

Description:

Slow HTTP attacks are denial-of-service (DoS) attacks in which the attacker sends HTTP requests in pieces slowly, one at a time to a Web server. If an HTTP request is not complete, or if the transfer rate is very low, the server keeps its resources busy waiting for the rest of the data. When the server’s concurrent connection pool reaches its maximum, this creates a DoS. Slow HTTP attacks are easy to execute because they require only minimal resources from the attacker.

Example:

<https://csye6225-fall2018-sanghavida.me//transaction>

Result:

1. Vulnerable to slow Http attacks on web application
2. Easy for attackers to execute attacks
3. Timeout after a period of waiting for acceptance from server

Solution:

1. Reject / drop connections with HTTP methods (verbs) not supported by the URL.
2. Limit the header and message body to a minimal reasonable length
3. Set an absolute connection timeout, if possible
4. Define the minimum incoming data rate, and drop connections that are slower than that rate
5. Make server backlog support reasonably large to handle small attacks

Reference:

https://blog.qualys.com/securitylabs/2011/11/02/how-to-protect-against-slow-http-attacks

**Conclusion:**

By installing AWS web application firewall and implementing SSL protocol, the application could be safe from most of the attacks