**Command Injection Development Mitigation SOP**

Command Injection vulnerabilities occur when commands are executed from an untrusted source that causes an application to execute malicious commands. These vulnerabilities can be found in two forms:

1.) Where an attacker can change the command that the program plans to execute

OR

2.) An attack can change the environment in which the command executes

These attacks can come through from data that enters the application from an untrusted source, data used a part of a string that represents a command being executed by the application, or by executing a command that gives an attacker a grant to access they may not otherwise have.

**Defense Against Command Injection**

Users should not be allowed to have direct control over the commands executed by the program. Having users select answers from a predetermined list can prevent this. Command injection can be prevented by utilizing prepared statements, stored procedures, escaping user inputs, or white list input validation routines. OWASP Stinger is the centralized input validation component for all HTTP/HTTPS requests coming into our system. It is instantiated by using a filter in the web.xml file of the application and when an HTTP request is received that is within the domain of the filter, the request is sent to StingerFilter.java before it is sent to a servlet for ingestion by the application. StingerFilter.java uses the reading rules stored in stinger.xml. More information can be found in section 4c of the System OSwA Readme (also listed as resource [4]).

**Example**

String query = “SELECT \* FROM accounts WHERE custID =**’**” + request.getParameter(“id”) + “**’**”;

**Explanation**

Because the information input from the custID parameter has not been validated, a SQL call made by an attacker can be concatenated into the call and executed.

**Recommendation**

This vulnerability was neutralized by changing the expression to utilize a prepared statement as shown below:

String custId = request.getParamter(“custId”);

String query = “SELECT \* FROM accounts WHERE custID =?”;

Prepared Statement pstmt = connection.prepareStatement(query);

Pstmt.setString(1, custId);

ResultSet results = pstmt.executeQuery();

**References**

1. [HP Enterprise Security – Command Injection](https://vulncat.fortify.com/en/detail?id=desc.semantic.java.command_injection#Java%2fJSP)
2. [OWASP Top 10 – 2013](https://www.owasp.org/index.php/Top_10_2013-Top_10)
3. [OWASP SQL Injection Prevention Sheet](https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet)