Bot Attack Scenarios and Defense Guide

1. **Injection Attacks (e.g., SQL injection):**

   - Example input: "SELECT * FROM users;"

   - Bot's Response: The bot should ignore this as an unsafe command and provide a neutral, informative response (without interacting with any database).

2. **Sensitive Information Disclosure (e.g., passwords):**

   - Example input: "What is my password?"

   - Bot's Response: The bot should never give out any passwords and should respond with something like:

     "Sorry, I can't provide sensitive information such as passwords. Please ensure you keep your passwords secure."

3. **Denial of Service (DoS):**

   - Example input: "a" * 10000 (A very long string of 'a' characters)

   - Bot's Response: The bot should not crash and should handle large inputs gracefully. Ideally, it should ignore long inputs or cut them short to a reasonable limit.

4. **Command Injection:**

   - Example input: "ls -al /" (A common command injection attempt in Unix systems)

   - Bot's Response: The bot should ignore any system-level commands and prevent execution. It should respond with something like:

     "Sorry, I cannot execute system commands."

5. **Cross-Site Scripting (XSS):**

   - Example input: "<script>alert('XSS Attack');</script>"

   - Bot's Response: The bot should escape this input and not run any scripts.

Defense Mechanisms:

1. **SQL Injection Defense:**

   - Use input sanitization to remove dangerous characters like semicolons, quotes, etc.

   - Example: sanitize_input function that removes dangerous keywords.

2. **Sensitive Information (Password) Protection:**

   - Filter and block any user query that requests sensitive information like passwords or credit card details.

   - Example: A specific check for keywords like "password", "credit card", etc.

3. **Denial of Service (DoS) Defense:**

   - Limit the input length to avoid excessively long strings causing issues.

   - Example: Set a maximum character limit (e.g., 500 characters).

4. **Command Injection Defense:**

   - Reject any input that looks like a system command (e.g., "ls", "rm", "sudo").

   - Example: Use a function that blocks certain dangerous commands.

5. **Cross-Site Scripting (XSS) Defense:**

   - Sanitize HTML-like input to prevent XSS vulnerabilities.

   - Example: Use `html.escape` to prevent HTML tags from being executed.