Detailed Steps to Solve the Machine

Machine Information

- **Macro**: WS
- **Type**: SQL Injection
- **Description**: The machine hosts a vulnerable PHP application with a login endpoint susceptible to SQL injection in the username parameter of a POST request. The application fails to sanitize user input, allowing manipulation of SQL queries, including UPDATE statements, to escalate privileges and access sensitive data. The flag is retrieved by exploiting this vulnerability to dump the credential table in the users database.
- **Objective**: Retrieve the flag by exploiting SQL injection to manipulate the database and extract sensitive data from the users database.

Step-by-Step Process

Step 1: Network Discovery with Nmap

- **Command**: nmap -sn 192.168.2.0/24
- **Description**:
    - **Purpose**: Perform a ping scan to identify live hosts on the 192.168.2.0/24 subnet.
    - **Details**:
        - Executed from a machine with IP 192.168.0.5.
        - nmap -sn conducts a host discovery scan without port scanning, checking which IPs in the 192.168.2.0/24 range (256 addresses) are active.
        - Identifies the target machine's IP address within the network.
    - **Assumption**: The scan reveals 192.168.2.4 as a live host, which we target in subsequent steps.
    - **Output**: A list of active IPs, including 192.168.2.4.

Step 2: Service Scanning with Nmap

- **Command**: nmap -sV 192.168.2.4

- **Description**:
    - **Purpose**: Identify open ports and services on the target machine (192.168.2.4).
    - **Details**:
        - nmap -sV performs a service version scan, detecting open ports and software versions.
        - Executed from 192.168.0.5.
        - Critical for identifying services like HTTP, implied by later curl commands.
    - **Assumption**: The scan reveals port 80 (HTTP) is open, running a PHP web application.
    - **Output**: A report listing open ports, with port 80 (HTTP) confirmed as the entry point.

Step 3: Access Web Application

- **Command**: curl http://192.168.2.4:80/index.html
- **Description**:
    - **Purpose**: Interact with the web application on port 80 to explore its functionality.
    - **Details**:
        - Executed from 192.168.0.5.
        - Sends an HTTP GET request to the root endpoint (index.html).
    - **Assumption**: The response indicates a PHP application with a login endpoint (login.php), suggesting potential for SQL injection vulnerabilities.
    - **Output**: HTML or text describing the web application, likely referencing login.php.

Step 4: Initial SQL Injection Test with sqlmap

- **Command**: sqlmap -u "http://192.168.2.4/login.php" --data="username=admin&password=admin" --dbs
- **Description**:

- **Purpose**: Use sqlmap to test the POST parameters for SQL injection and enumerate available databases.
- **Details**:
  - Executed from 192.168.0.5.
  - sqlmap targets the username and password parameters in the POST request to login.php.
  - --dbs enumerates the database names accessible via the injection point.
- **Assumption**: The tool confirms SQL injection vulnerability and identifies a database named users.
- **Output**: A list of databases, including users.

Step 5: Enumerate Tables in users Database

- **Command**: sqlmap -u "http://192.168.2.4/login.php" --data="username=admin&password=admin" -D users --tables
- **Description**:
  - **Purpose**: Identify tables within the users database.
  - **Details**:
    - Targets the users database with the same injection point.
    - --tables lists all tables in the specified database.
  - **Assumption**: The output reveals a table named credential.
  - **Output**: A list of tables, including credential.

Step 6: Enumerate Columns in credential Table

- **Command**: sqlmap -u "http://192.168.2.4/login.php" --data="username=admin&password=admin" -D users -T credential --columns
- **Description**:
  - **Purpose**: Identify the columns in the credential table to locate sensitive data.
  - **Details**:

        - Targets the credential table in users.
        - --columns enumerates the column names.
    - **Assumption**: The output includes columns such as username, password, and isAdmin, indicating potential privilege-related data.
    - **Output**: A list of columns, including username, password, and isAdmin.

Step 7: Dump the credential Table

- **Command**: sqlmap -u "http://192.168.2.4/login.php" --data="username=admin&password=admin" -D users -T credential --dump
- **Description**:
    - **Purpose**: Extract the contents of the credential table to identify sensitive data.
    - **Details**:
        - Dumps all data in the credential table.
        - Executed from 192.168.0.5.
    - **Assumption**: The dumped data reveals user credentials but may not directly contain the flag, suggesting further exploitation is needed.
    - **Output**: A table of credentials, including usernames, passwords, and isAdmin values.

Step 8: Exploit SQL Injection with UPDATE Statement

- **Command**: curl -v -L --cookie cookie.txt --cookie-jar cookie.txt 'http://192.168.2.4/login.php' --data-raw 'username=student%27%3Bupdate+credential+set+isAdmin%3D1%3B--&password=student'
- **Description**:
    - **Purpose**: Exploit SQL injection in the username parameter to execute an UPDATE statement, escalating the student account to admin privileges.
    - **Details**:
        - Sends a POST request to login.php with a malicious username: student';update credential set isAdmin=1;--.

        - The injection closes the SQL query with a single quote ('), appends an UPDATE statement to set isAdmin=1 for all records in the credential table, and comments out the rest of the query with --.
        - --cookie and --cookie-jar manage session cookies to maintain state.
        - Executed from 192.168.0.5.
    - **Assumption**: The injection succeeds, updating the isAdmin column, granting admin access to the student account.
    - **Output**: A response indicating successful login or redirection, confirming the privilege escalation.

Step 9: Retrieve the Flag

- **Description**:
    - **Purpose**: Access the flag after gaining admin privileges.
    - **Details**:
        - With isAdmin=1, the student account can access a restricted admin area (assumed to be part of login.php or a related endpoint).
        - The flag is retrieved from this area, likely displayed or accessible via a subsequent request.
    - **Assumption**: The flag babnDtSmpXsCrtdW is obtained after logging in with the escalated student account.
    - **Output**: The flag: babnDtSmpXsCrtdW.

Final Answer

- **Flag**: babnDtSmpXsCrtdW