Detailed Steps to Solve the Machine

Machine Information
• **Macro**: WS
• **Type**: Remote Code Execution (RCE)
• **Description**: The machine hosts a vulnerable PHP web application with an 'image' parameter that allows arbitrary PHP code execution. The application fails to sanitize input, enabling attackers to inject and execute malicious PHP code, such as calling phpinfo() or executing system commands to read the /flag.txt file. The flag is retrieved by exploiting this vulnerability.
• **Objective**: Retrieve the flag by exploiting the 'image' parameter to achieve remote code execution and access /flag.txt.

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 192.168.2.6 as a live host, targeted in subsequent steps.
- **Output**: A list of active IPs, including 192.168.2.6.

Step 2: Service Scanning with Nmap
• **Command**: nmap -sV 192.168.2.6
• **Description**:

- **Purpose**: Identify open ports and services on the target machine (192.168.2.6).
- **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://192.168.2.6
• **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.
- **Assumption**: The response indicates a PHP application with a vulnerable 'image' parameter, suggesting potential for code injection.
- **Output**: HTML or text describing the web application, likely referencing a page that processes the 'image' parameter.

Step 4: Test PHP Code Execution
• **Command**: curl "http://192.168.2.6/?image=\">'phpinfo();//"
• **Description**:

- **Purpose**: Test the 'image' parameter for PHP code execution by injecting a phpinfo() call.
- **Details**:
  - Executed from 192.168.0.5.
  - The URL-encoded payload \">';phpinfo();// injects PHP code to execute phpinfo(), revealing server configuration.
  - The payload escapes the expected input format and appends executable PHP code.
- **Assumption**: The response includes the phpinfo() output, confirming the ability to execute arbitrary PHP code.

- **Output**: A page displaying PHP configuration details, indicating successful code execution.

Step 5: Retrieve the Flag via RCE
• **Command**: curl "http://192.168.2.6/?image=\">';system(\"cat%20/flag.txt\");//"
• **Description**:

- **Purpose**: Execute a system command via the 'image' parameter to read the contents of /flag.txt.
- **Details**:
    - Sends a GET request with the payload \">';system(\"cat /flag.txt\");//, which executes the cat /flag.txt command on the server.
    - %20 is the URL-encoded space character.
    - The system() function runs the command, and the output is returned in the response.
- **Assumption**: The response contains the flag feraB7zisg5g5rvT.
- **Output**: The flag: feraB7zisg5g5rvT.

Final Answer
• **Flag**: feraB7zisg5g5rvT