
Detailed Steps to Solve the Machine

Machine Information

- **Macro**: WS
- **Type**: Path Traversal
- **Description**: The machine hosts a vulnerable PHP application with naive path traversal filters that can be bypassed using encoded relative paths. This allows access to sensitive files outside the web root. The flag is retrieved by exploiting this vulnerability to read /root/flag.txt.
- **Objective**: Retrieve the flag by bypassing the naive path traversal filters in the PHP application to access /root/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 192.168.2.2 as a live host, which we target in subsequent steps.
  - **Output**: A list of active IPs, including 192.168.2.2.

Step 2: Service Scanning with Nmap

- **Command**: nmap -sV 192.168.2.2
- **Description**:
    - **Purpose**: Identify open ports and services on the target machine (192.168.2.2).
    - **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://192.168.2.2:80/index.php
- **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.php).
    - **Assumption**: The response indicates a PHP application with a page parameter, suggesting potential for path traversal vulnerabilities, though with naive filtering mechanisms.
    - **Output**: HTML or text describing the web application, likely a simple page rendered by index.php.

Step 4: Test Path Traversal with Sensitive File

- **Command**: curl http://192.168.2.2:80/index.php?page=.../[./.../](./.../)./etc/passwd

- **Description**:
  - **Purpose**: Test the PHP application for path traversal vulnerabilities by bypassing naive filters to access a sensitive system file.
  - **Details**:
    - Sends a GET request with the page parameter set to .../.../.../.../.../.../etc/passwd.
    - The pattern .../.../ is used to bypass simplistic filters that block standard ../ sequences, as the application may fail to normalize or properly sanitize this encoded relative path.
    - Exploits the application's vulnerability to access /etc/passwd.
  - **Assumption**: The response contains the contents of /etc/passwd, confirming the path traversal vulnerability and the ability to bypass naive filters.
  - **Output**: Contents of /etc/passwd, indicating the application is vulnerable to encoded path traversal.

Step 5: Retrieve the Flag

- **Command**: curl http://192.168.2.2:80/index.php?page=.../.../.../.../.../.../root/flag.txt
- **Description**:
  - **Purpose**: Exploit the encoded path traversal vulnerability to read the flag file located at /root/flag.txt.
  - **Details**:
    - Sends a GET request with the page parameter set to .../.../.../.../.../.../root/flag.txt.
    - The application includes the contents of /root/flag.txt in the response due to the bypass of naive path traversal filters.
  - **Assumption**: The response contains the flag bHN6WEm1IERIy8Oc.
  - **Output**: The flag: bHN6WEm1IERIy8Oc.

Final Answer

- **Flag**: bHN6WEm1IERIy8Oc