
Detailed Steps to Solve the Machine

Machine Information

Name: CRPT
Type: Brute-force, Diffie-Hellman with short private key
Description: The machine involves exploiting a web service running on port 8080, performing network reconnaissance, and potentially leveraging a weak cryptographic implementation (Diffie-Hellman with a short private key) to gain access to sensitive data or escalate privileges.
Objective: Gain root access to the target machine and retrieve the flag located at /root/flag.

Step-by-Step Process

Step 1: Network Discovery with Nmap
Command: nmap -sn 192.168.4.0/24
Description:

- **Purpose**: Perform a ping scan to identify live hosts on the 192.168.4.0/24 subnet.
- **Details**:
    - The command is executed from a machine with IP 192.168.0.5.
    - nmap -sn conducts a host discovery scan (ping scan) without port scanning, checking which IP addresses in the 192.168.4.0/24 range (256 addresses) are active.
    - This step identifies the target machine's IP address within the network.
- **Assumption**: The scan reveals that 192.168.4.3 is a live host, which will be targeted in subsequent steps.
- **Output**: A list of active IP addresses, including 192.168.4.3.

Step 2: Port Scanning with Nmap
Command: nmap -sS 192.168.4.3
Description:

- **Purpose**: Identify open ports and services running on the target machine (192.168.4.3).
- **Details**:
    - nmap -sS performs a stealth SYN scan to detect open ports without completing TCP connections, minimizing detection.
    - Executed from 192.168.0.5, targeting 192.168.4.3.
    - This step is critical to identify services, such as a web server, that can be exploited.
- **Assumption**: The scan reveals that port 8080 is open, running an HTTP service.
- **Output**: A report listing open ports, with port 8080 (HTTP) confirmed as a potential entry point.

Step 3: Web Service Enumeration with Curl
Command: curl http://192.168.4.3:8080
Description:

- **Purpose**: Access the web service running on port 8080 to gather initial information.
- **Details**:
    - curl retrieves the content of the HTTP service at http://192.168.4.3:8080.
    - Executed from 192.168.0.5.
    - This step helps understand the web application's functionality, potentially revealing a login page, API endpoints, or cryptographic operations.

- **Assumption**: The response contains a webpage or API that hints at a Diffie-Hellman key exchange or related cryptographic functionality.
- **Output**: The HTML or JSON response from the web server, providing clues about the service.

Step 4: Exploring Web Endpoints
Command: curl http://192.168.4.3:8080/source
Description:

- **Purpose**: Retrieve the source code or additional information from the /source endpoint.
- **Details**:
    - The /source endpoint may expose the application's source code or configuration details, which could reveal vulnerabilities.
    - Executed from 192.168.0.5.
    - This aligns with the machine's theme of exploiting a weak Diffie-Hellman implementation.
- **Assumption**: The response contains source code or parameters (e.g., prime number, generator, or public keys) used in a Diffie-Hellman key exchange with a short private key.
- **Output**: Source code or cryptographic parameters that can be analyzed for vulnerabilities.

Step 5: Accessing Output Endpoint
Command: curl http://192.168.4.3:8080/output
Description:

- **Purpose**: Retrieve data from the /output endpoint, potentially containing encrypted data or a flag.
- **Details**:
    - The /output endpoint may provide the result of a cryptographic operation or sensitive data.

    - Executed from 192.168.0.5.
    - This step may confirm the need to break the Diffie-Hellman key exchange to access protected resources.
- **Assumption**: The response is encrypted or inaccessible without exploiting the weak Diffie-Hellman implementation.
- **Output**: Encrypted data or an error message indicating restricted access.

Step 6: Exploit Development
Command: WriteFile(content=fname, file_name='solution.py')
Description:

- **Purpose**: Create a Python script to exploit the weak Diffie-Hellman implementation.
- **Details**:
    - The solution.py script is written to perform cryptanalysis, likely computing discrete logarithms to recover the short private key used in the Diffie-Hellman exchange.
    - The content (fname) is assumed to be a Python script tailored to the parameters obtained from the /source endpoint.
    - This step leverages the machine's vulnerability: a Diffie-Hellman implementation with a short private key, which is computationally feasible to brute-force.
- **Assumption**: The script calculates the shared secret, decrypts the /output data, or gains privileged access to the system.
- **Output**: A file named solution.py stored on the attacking machine (192.168.0.5).

Step 7: Execute Exploit
Command: python3 /root/scripts/solution.py
Description:

- **Purpose**: Run the solution.py script to exploit the vulnerability and gain access.

- **Details**:
  - The script is executed on the attacking machine (192.168.0.5).
  - It likely recovers the private key, computes the shared secret, and either decrypts the flag or provides a method to escalate privileges (e.g., by accessing a restricted endpoint or gaining a shell).
- **Assumption**: The script's execution grants access to the flag or a shell on the target machine (192.168.4.3).
- **Output**: The flag, a shell, or intermediate data required to complete the objective.

Step 8: Retrieve the Flag
Description:

- **Purpose**: Obtain the final flag to complete the objective.
- **Details**:
  - Based on the exploit's success, the flag is either retrieved directly from the decrypted /output data or by gaining a shell and accessing /root/flag.
  - If a shell is obtained, the command cat /root/flag would be used to read the flag.
- **Output**: The flag: QSiAdm6quEZk7B90.

Final Answer

Flag: QSiAdm6quEZk7B90