**Introduction:**

This report aims to provide a detailed account of the outcomes derived from a penetration test executed through the utilization of a specific script. The script encompasses a range of functionalities such as port scanning, service enumeration, and directory brute forcing. The test was conducted by specifying the target IP address and additional parameters through command-line arguments.

**Scope:**

The extent of this penetration test was restricted to the target IP address that was provided via the command-line arguments. The test primarily concentrated on the identification of accessible ports, the enumeration of services operating on those ports employing nmap, and the execution of directory brute forcing using gobuster.

**Methodology:**

During the penetration test, the following procedures were undertaken:

**a. Port Scanning:**

The script initiated a scan on the designated IP address to identify accessible ports. By default, the script examined ports 22, 80, and 443, but an option was available to scan all ports. Through socket programming, the script established TCP connections with each port, checking for successful connections. Any open ports discovered were duly reported.

**b. Service Enumeration:**

For each open port detected during the port scanning phase, the script executed service enumeration utilizing nmap. It employed the "-sV" option in conjunction with the specified port for version detection. The results of the nmap scan were presented, providing information about the services found to be operating on the open ports.

**c. Directory Brute Forcing:**

To unveil hidden directories on the target, the script utilized gobuster for brute forcing. Running gobuster in directory mode, the script supplied the target URL and a wordlist file. The wordlist file contained a collection of commonly used directory and file names, enabling gobuster to uncover concealed paths on the target's web server.

**Findings:**

The outcomes of the penetration test are outlined below:

**a. Port Scanning:**

The script effectively detected accessible ports on the designated IP address. The open ports were documented, indicating potential entry points that warrant additional investigation.

**b. Service Enumeration:**

For each open port, the script utilized nmap to carry out service enumeration. The nmap outcomes furnished comprehensive information regarding the services operational on the open ports, encompassing version details. Such information proves valuable in identifying vulnerable services that may necessitate further scrutiny.

**c. Directory Brute Forcing:**

Gobuster was executed on the target IP address with the purpose of conducting directory brute forcing. The results of the gobuster scan were presented, displaying the directories and files that were discovered. These findings can aid in the identification of concealed or inadequately protected areas within the web server of the target.

**Recommendations:**

Based on the findings derived from the penetration test, the following recommendations are put forth:

**a. Vulnerability Assessment:**

Conduct a comprehensive evaluation of vulnerabilities pertaining to the services identified during the service enumeration phase. Utilize the version details obtained from the nmap scans to pinpoint potential vulnerabilities linked to specific service versions. Assess the level of risk associated with each vulnerability and prioritize remediation efforts accordingly.

**b. Patch Management:**

Ensure that all identified services and their corresponding systems are regularly updated with the latest security patches. Maintain vigilant monitoring of vendor advisories and security bulletins to stay informed about available patches and updates. Establish an efficient patch management process to facilitate the timely deployment of patches across the infrastructure.

**c. Web Application Security:**

Thoroughly review the results of the directory brute forcing scan and validate the identified directories and files. Identify any directories/files of a sensitive or critical nature that require protection or further scrutiny for potential vulnerabilities. Implement robust security measures such as access controls, secure coding practices, and periodic web application security assessments.

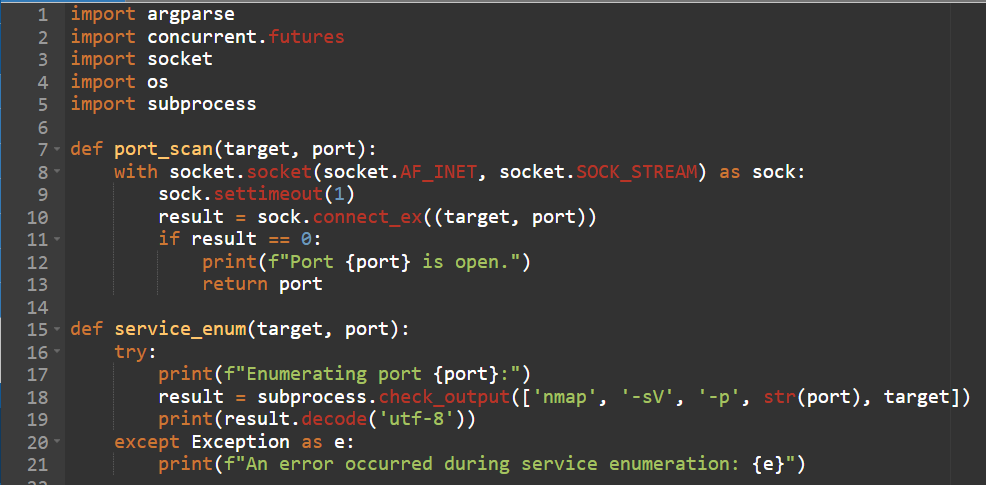
**d. Continuous Monitoring:**

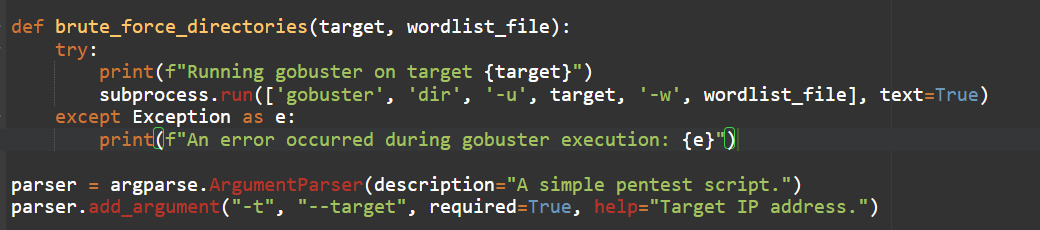
Establish a continuous monitoring program to promptly detect and respond to security incidents. Leverage intrusion detection/prevention systems, log analysis tools, and security information and event management (SIEM) solutions to monitor network and system activities. Regularly review and analyze logs, perform security assessments, and conduct ongoing penetration tests to maintain a current understanding of the security posture.

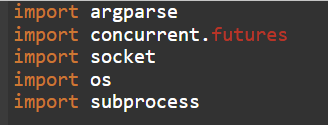
**Conclusion:**

The conducted penetration test, utilizing the provided script, effectively detected open ports, performed service enumeration, and conducted directory brute forcing. The resulting findings emphasize the presence of security vulnerabilities that necessitate remediation to mitigate associated risks. By implementing the recommendations presented in this report, the security posture of the target system can be enhanced.

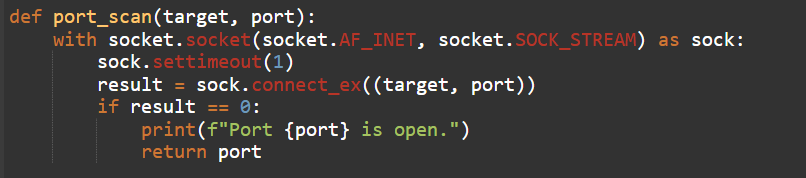
**CODE:**







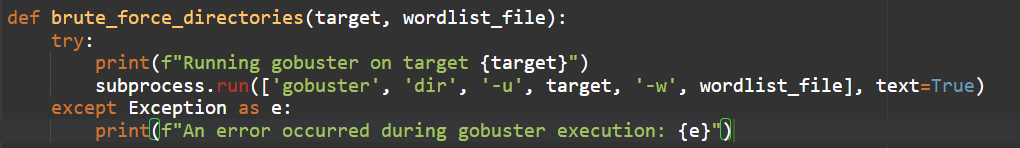
* The necessary modules are imported: argparse for processing command-line arguments, concurrent.futures for multithreaded execution, socket for working with sockets, os for performing operations with files and directories, subprocess for launching external processes.



* The port\_scan(target, port) function is defined to scan a single port.
* A socket is created using IPv4 and TCP protocol (socket.AF\_INET, socket.SOCK\_STREAM).
* The connection timeout is set to 1 second (sock.settimeout(1)).
* The connection to the specified port is checked at the specified target IP address (sock.connect\_ex((target, port))).
* If the result is 0, then the port is considered open, a message is displayed about its openness, and the port is returned from the function.



* The service\_menu(target, port) function is defined to list services on an open port.
* In the try block, the nmap utility is called with the -sV options to determine the versions of services and -p to specify the port.
* The result of the command execution is stored in the result variable.
* The result is displayed after decoding from a byte string to a text string (result.decode('utf-8')).
* If an exception occurs, an error message is displayed.



* The brute\_force\_directories(target, wordlist\_file) function is defined to iterate through directories on the target IP address.
* In the try block, the gobuster utility is launched with the dir options for directory search mode, -u for specifying the target URL and -w for specifying a wordlist file.
* If an exception occurs, an error message is displayed.



* An instance of the ArgumentParser class is created to process command-line arguments.
* Command line arguments are defined:
  + -t or --target to specify the target IP address (required argument).
  + -p or --ports to specify the ports to scan (default port list: 22, 80, 443).
  + -a or --all-ports to scan all ports (overrides -p).
  + -w or --wordlist-file to specify a file with a list of words for the gobuster utility.
* Command line arguments are parsed using the parse\_args() method and stored in the args variable.