Task 3: PENETRATION TESTING TOOLKIT

Instruction:

BUILD A TOOLKIT WITH MULTIPLE MODULES (E.G., PORT SCANNER, BRUTE-FORCER) FOR PENETRATION TESTING.

DELIVERABLE:

A PYTHON-BASED MODULAR TOOLKIT WITH DETAILED DOCUMENTATION

Script:

```
import socket
import threading
import itertools
import hashlib
import requests
from concurrent.futures import ThreadPoolExecutor

class PenTestToolkit:
    def __init__(self):
        self.target = None

    def set_target(self, target):
        self.target = target

    def port_scanner(self, start_port, end_port):
        open_ports = []
        def scan_port(port):
```

```
            sock = socket.socket(socket.AF_INET, socket.SOCK_STR
            sock.settimeout(1)
            result = sock.connect_ex((self.target, port))
            if result == 0:
                open_ports.append(port)
            sock.close()

        with ThreadPoolExecutor(max_workers=100) as executor:
            executor.map(scan_port, range(start_port, end_port +

        return open_ports

    def brute_forcer(self, url, username, wordlist_path, success
        def try_password(password):
            response = requests.post(url, data={"username": user
            return success_message not in response.text

        with open(wordlist_path, 'r') as f:
            passwords = f.read().splitlines()

        for password in passwords:
            if try_password(password):
                return password

        return None

    def hash_cracker(self, hash_to_crack, wordlist_path, hash_ty
        def hash_password(password):
            if hash_type == 'md5':
                return hashlib.md5(password.encode()).hexdigest(
            elif hash_type == 'sha1':
                return hashlib.sha1(password.encode()).hexdigest
            elif hash_type == 'sha256':
                return hashlib.sha256(password.encode()).hexdige

        with open(wordlist_path, 'r') as f:
```

```
            passwords = f.read().splitlines()

        for password in passwords:
            if hash_password(password) == hash_to_crack:
                return password

        return None

# Usage example
toolkit = PenTestToolkit()

# Port Scanner
toolkit.set_target('example.com')
open_ports = toolkit.port_scanner(1, 1000)
print(f"Open ports: {open_ports}")

# Brute Forcer
password = toolkit.brute_forcer('http://example.com/login', 'adr
print(f"Cracked password: {password}")

# Hash Cracker
cracked_password = toolkit.hash_cracker('5f4dcc3b5aa765d61d8327
print(f"Cracked hash: {cracked_password}")
```

This modular toolkit includes three main components:

Port Scanner

The `port_scanner` method scans a range of ports on the specified target using multi-threading for improved performance. It returns a list of open ports.

Brute Forcer

The `brute_forcer` method attempts to crack a login form by trying passwords from a wordlist. It sends POST requests to the specified URL and checks for a success

message in the response.

Hash Cracker

The `hash_cracker` method attempts to crack a given hash using a wordlist. It supports MD5, SHA1, and SHA256 hash types.

Usage

1. Create an instance of the `PenTestToolkit` class.
2. Use the `set_target` method to specify the target for port scanning.
3. Call the desired method with appropriate parameters.