**Task – 3 :- Penetration\_Toolkit**

**port\_scanning**

import socket

import concurrent.futures

def scan\_port(target, port):

try:

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

s.settimeout(1)

s.connect((target, port))

try:

banner = s.recv(1024).decode(errors="ignore").strip()

except:

banner = "No banner"

try:

service = socket.getservbyport(port, "tcp")

except:

service = "unknown"

return (port, service, banner)

except:

return None

def scan\_ports(target, ports=range(1, 65536)): # full port range

open\_ports = []

print(f"[\*] Full scan started on {target}... This may take some time.")

with concurrent.futures.ThreadPoolExecutor(max\_workers=500) as executor:

results = executor.map(lambda p: scan\_port(target, p), ports)

for result in results:

if result:

open\_ports.append(result)

return open\_ports

**bruteforce.py**

import requests

def brute\_force(url, usernames, passwords):

for user in usernames:

for pwd in passwords:

response = requests.post(url, data={"username": user, "password": pwd})

if "Welcome" in response.text:

return f"[+] Valid credentials found: {user}:{pwd}"

return "[-] No valid credentials found"

**host\_discovery.py**

from scapy.all import ARP, Ether, srp, ICMP, IP, sr1, conf, get\_if\_addr, get\_working\_if

import ipaddress

import subprocess

def get\_local\_subnet():

iface = get\_working\_if()

ip = get\_if\_addr(iface)

subnet = ip.split('.')[:-1]

return iface, '.'.join(subnet) + '.0/24'

def arp\_scan(subnet, iface):

print("[\*] Scanning with ARP...")

arp = ARP(pdst=subnet)

ether = Ether(dst="ff:ff:ff:ff:ff:ff")

packet = ether / arp

result = srp(packet, timeout=2, iface=iface, verbose=False)[0]

return [(rcv.psrc, rcv.hwsrc) for \_, rcv in result]

def icmp\_scan(subnet):

print("[\*] Scanning with ICMP...")

live\_hosts = []

for ip in ipaddress.IPv4Network(subnet):

pkt = IP(dst=str(ip))/ICMP()

reply = sr1(pkt, timeout=1, verbose=False)

if reply:

live\_hosts.append((str(ip), 'ICMP response'))

return live\_hosts

def fallback\_nmap(subnet):

print("[!] Fallback to Nmap ping scan...")

try:

output = subprocess.check\_output(['nmap', '-sn', subnet], stderr=subprocess.STDOUT, universal\_newlines=True)

hosts = []

ip = None

for line in output.splitlines():

if "Nmap scan report for" in line:

ip = line.split()[-1]

if "MAC Address:" in line and ip:

mac = line.split("MAC Address: ")[1].split()[0]

hosts.append((ip, mac))

ip = None

elif ip:

hosts.append((ip, "Unknown"))

ip = None

return hosts

except Exception as e:

return [("Error", f"Nmap failed: {e}")]

def discover\_network(subnet=None):

iface, default\_subnet = get\_local\_subnet()

if subnet is None:

subnet = default\_subnet

print(f"[\*] Using interface: {iface}")

print(f"[\*] Target Subnet: {subnet}")

# Try ARP

arp\_hosts = arp\_scan(subnet, iface)

if arp\_hosts:

return arp\_hosts

# Try ICMP

icmp\_hosts = icmp\_scan(subnet)

if icmp\_hosts:

return icmp\_hosts

# Fallback to Nmap

nmap\_hosts = fallback\_nmap(subnet)

return nmap\_hosts

**service\_detector.py**

import socket

import ssl

import subprocess

def detect\_service(target, port):

try:

# Step 1: Try plain TCP or SSL socket connection

sock = socket.create\_connection((target, port), timeout=4)

# Step 2: Determine if port is HTTPS

if port in [443, 8443, 10443]:

try:

context = ssl.create\_default\_context()

ssl\_sock = context.wrap\_socket(sock, server\_hostname=target)

ssl\_sock.sendall(b"HEAD / HTTP/1.1\r\nHost: " + target.encode() + b"\r\n\r\n")

banner = ssl\_sock.recv(1024).decode(errors="ignore")

ssl\_sock.close()

if banner.strip():

return f"[+] SSL Banner:\n{banner.strip()}"

else:

return "[~] SSL connected, but no banner received."

except ssl.SSLError as e:

return f"[-] SSL handshake failed: {e}"

else:

# Step 3: HTTP service check (port 80 or others)

sock.sendall(b"HEAD / HTTP/1.0\r\n\r\n")

banner = sock.recv(1024).decode(errors="ignore")

sock.close()

return f"[+] TCP Banner:\n{banner.strip()}" if banner.strip() else "[~] No banner received."

except Exception as socket\_error:

# Step 4: Fallback to Nmap

try:

print("[!] Socket failed. Falling back to Nmap service detection...")

result = subprocess.check\_output(

["nmap", "-sV", "-p", str(port), target],

stderr=subprocess.STDOUT,

universal\_newlines=True,

)

return f"[+] Nmap fallback result:\n{result}"

except Exception as nmap\_error:

return f"[-] Both socket and Nmap failed: {nmap\_error}"

**main.py**

import os

import sys

from port\_scanner import scan\_ports

from bruteforce import brute\_force

from host\_discovery import discover\_network

from service\_detector import detect\_service

def check\_root():

if os.geteuid() != 0:

print("\n[-] ERROR: This script must be run as root (try: sudo python3 main.py)")

sys.exit(1)

def menu():

print("\n=== Penetration Testing Toolkit ===")

print("1. Port Scanner")

print("2. Brute-Force Login")

print("3. Host Discovery")

print("4. Service Detection")

print("0. Exit")

return input("Choose a module: ")

def main():

check\_root()

while True:

choice = menu()

if choice == "1":

target = input("Target IP: ").strip()

open\_ports = scan\_ports(target)

if open\_ports:

print(f"\n[+] Open ports on {target}:\n")

for port, service, banner in open\_ports:

print(f" - Port {port} ({service}): {banner}")

with open("scan\_results.txt", "w") as f:

for port, service, banner in open\_ports:

f.write(f"Port {port} ({service}): {banner}\n")

print("\n[+] Results saved to scan\_results.txt")

else:

print("[-] No open ports found.")

elif choice == "2":

url = input("Login URL (e.g., http://localhost/login): ").strip()

usernames = ["admin", "user"]

passwords = ["admin", "1234", "password"]

print("[\*] Launching brute-force attack...")

result = brute\_force(url, usernames, passwords)

print(result)

elif choice == "3":

subnet = input("Network Range (e.g., 192.168.1.0/24): ").strip()

print("[\*] Scanning for live hosts...")

hosts = discover\_network(subnet)

if hosts:

for ip, mac in hosts:

print(f"[+] {ip} => {mac}")

else:

print("[-] No hosts found in the specified range.")

elif choice == "4":

target = input("Target IP: ").strip()

try:

port = int(input("Port (e.g., 80): ").strip())

banner = detect\_service(target, port)

print(f"[+] Service banner: {banner}")

except ValueError:

print("[-] Invalid port. Please enter a numeric value.")

elif choice == "0":

print("Exiting... Stay sharp. 🐉")

break

else:

print("[-] Invalid choice. Please select from the menu.")

if \_\_name\_\_ == "\_\_main\_\_":

main()







