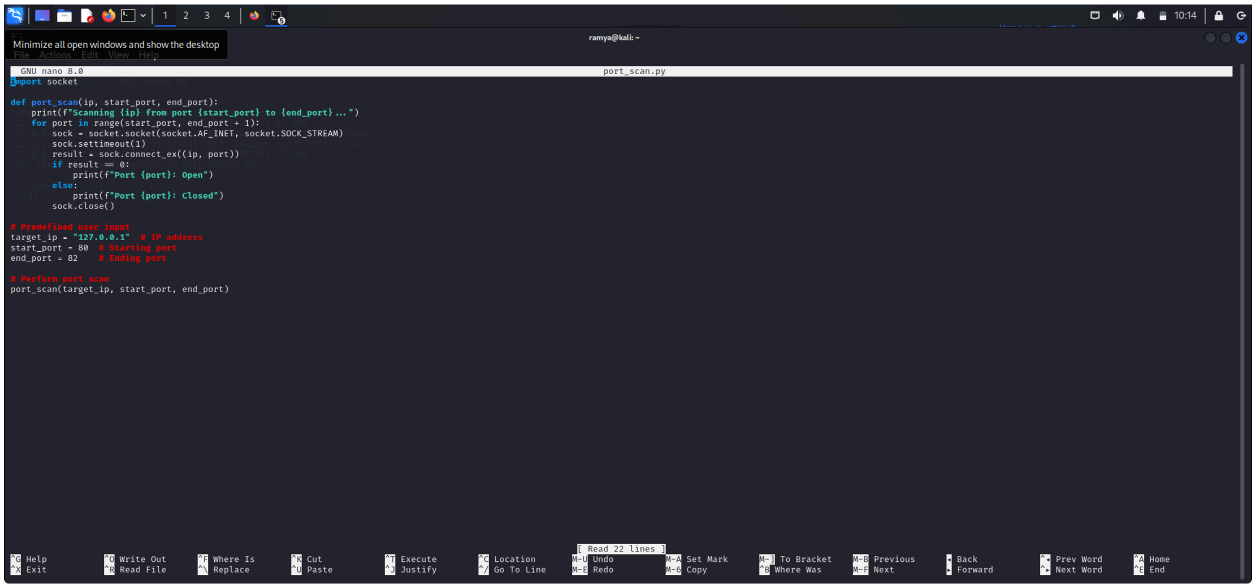
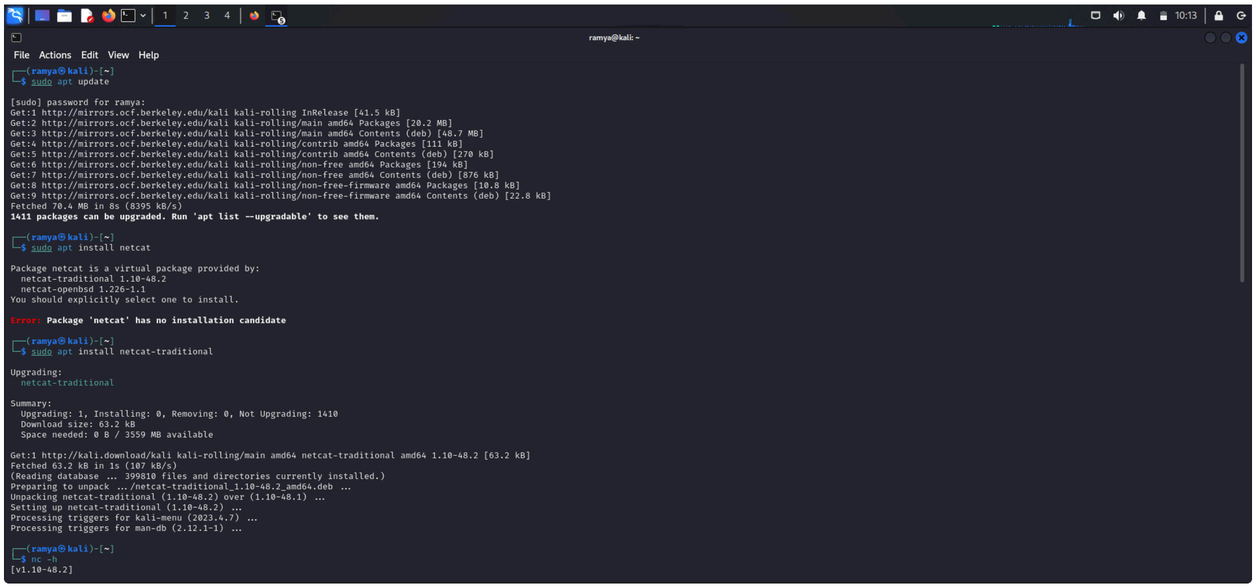
Bakka Ramyasree

Simple TCP Port Scanner

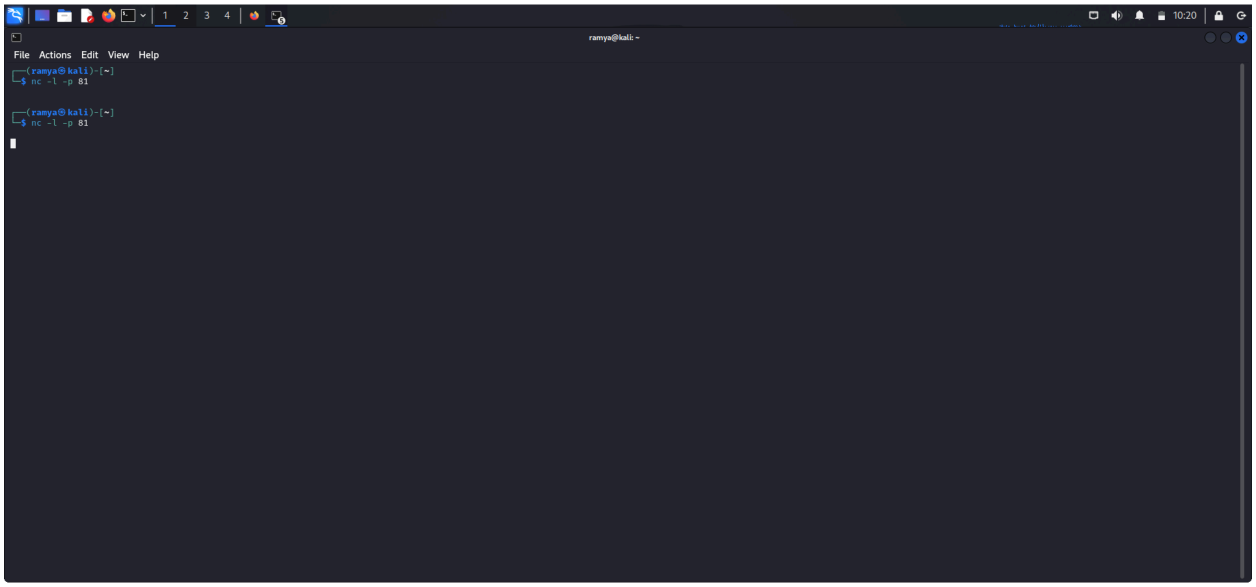
- Create a Python program that asks the user to input a target IP address and a range of ports.
- The program will attempt to connect to each port using the TCP protocol.
- If the connection is successful, the port is considered "open"; otherwise, it is "closed."



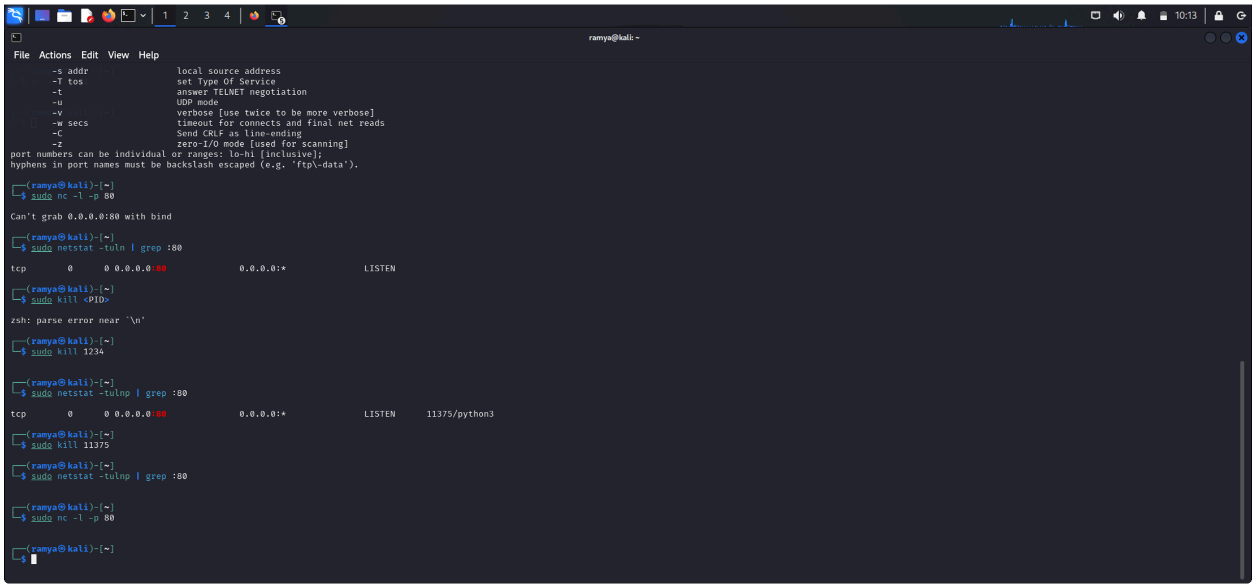
```
import socket

def port_scan(ip, start_port, end_port):
    print(f"Scanning {ip} from port {start_port} to {end_port}...")
    for port in range(start_port, end_port + 1):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(1)
        result = sock.connect_ex((ip, port))
        if result == 0:
            print(f"Port {port}: Open")
        else:
            print(f"Port {port}: Closed")
        sock.close()

# Predefined user input
target_ip = "127.0.0.1"  # IP address
start_port = 80  # Starting port
end_port = 82    # Ending port

# Perform port scan
port_scan(target_ip, start_port, end_port)
```



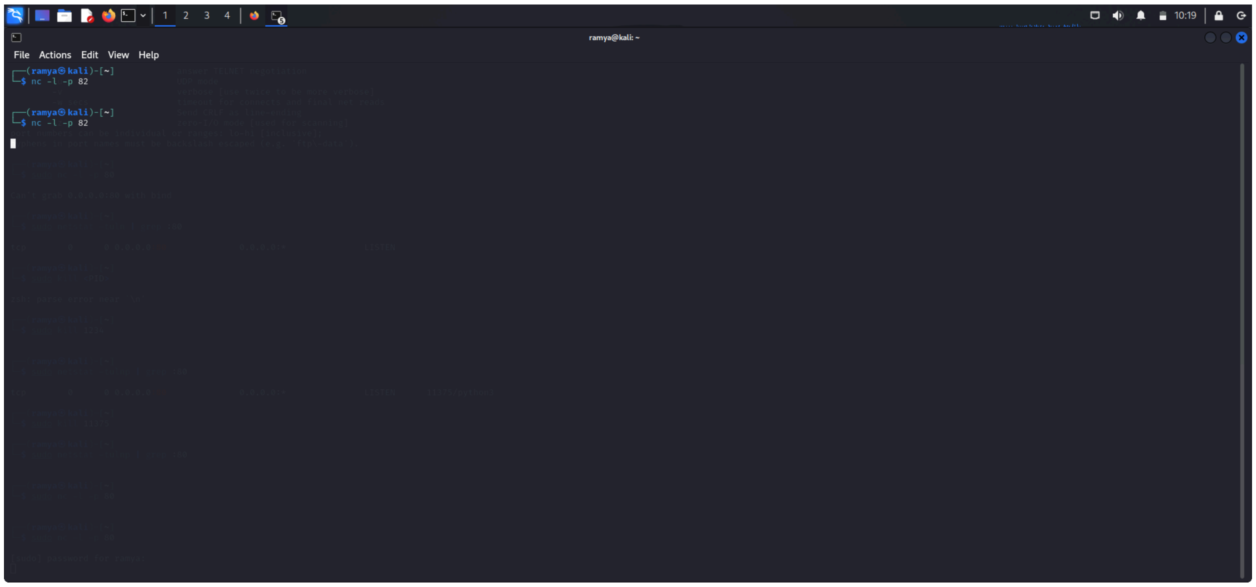
```
┌──(ramya㉿kali)-[~]
└─$ sudo apt update

[sudo] password for ramya:
Get:1 http://mirrors.ocf.berkeley.edu/kali kali-rolling InRelease [41.5 kB]
Get:2 http://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 Packages [20.2 MB]
Get:3 http://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 Contents (deb) [48.7 MB]
Get:4 http://mirrors.ocf.berkeley.edu/kali kali-rolling/contrib amd64 Packages [111 kB]
Get:5 http://mirrors.ocf.berkeley.edu/kali kali-rolling/contrib amd64 Contents (deb) [270 kB]
Get:6 http://mirrors.ocf.berkeley.edu/kali kali-rolling/non-free amd64 Packages [194 kB]
Get:7 http://mirrors.ocf.berkeley.edu/kali kali-rolling/non-free amd64 Contents (deb) [876 kB]
Get:8 http://mirrors.ocf.berkeley.edu/kali kali-rolling/non-free-firmware amd64 Packages [10.8 kB]
Get:9 http://mirrors.ocf.berkeley.edu/kali kali-rolling/non-free-firmware amd64 Contents (deb) [22.8 kB]
Fetched 70.4 MB in 8s (8395 kB/s)
1411 packages can be upgraded. Run 'apt list --upgradable' to see them.

┌──(ramya㉿kali)-[~]
└─$ sudo apt install netcat

Package netcat is a virtual package provided by:
  netcat-traditional 1.10-48.2
  netcat-openbsd 1.226-1.1
You should explicitly select one to install.

Error: Package 'netcat' has no installation candidate

┌──(ramya㉿kali)-[~]
└─$ sudo apt install netcat-traditional

Upgrading:
  netcat-traditional

Summary:
  Upgrading: 1, Installing: 0, Removing: 0, Not Upgrading: 1410
  Download size: 63.2 kB
  Space needed: 0 B / 3559 MB available

Get:1 http://kali.download/kali kali-rolling/main amd64 netcat-traditional amd64 1.10-48.2 [63.2 kB]
Fetched 63.2 kB in 1s (107 kB/s)
(Reading database ... 399810 files and directories currently installed.)
Preparing to unpack .../netcat-traditional_1.10-48.2_amd64.deb ...
Unpacking netcat-traditional (1.10-48.2) over (1.10-48.1) ...
Setting up netcat-traditional (1.10-48.2) ...
Processing triggers for kali-menu (2023.4.7) ...
Processing triggers for man-db (2.12.1-1) ...

┌──(ramya㉿kali)-[~]
└─$ nc -h
[v1.10-48.2]
```



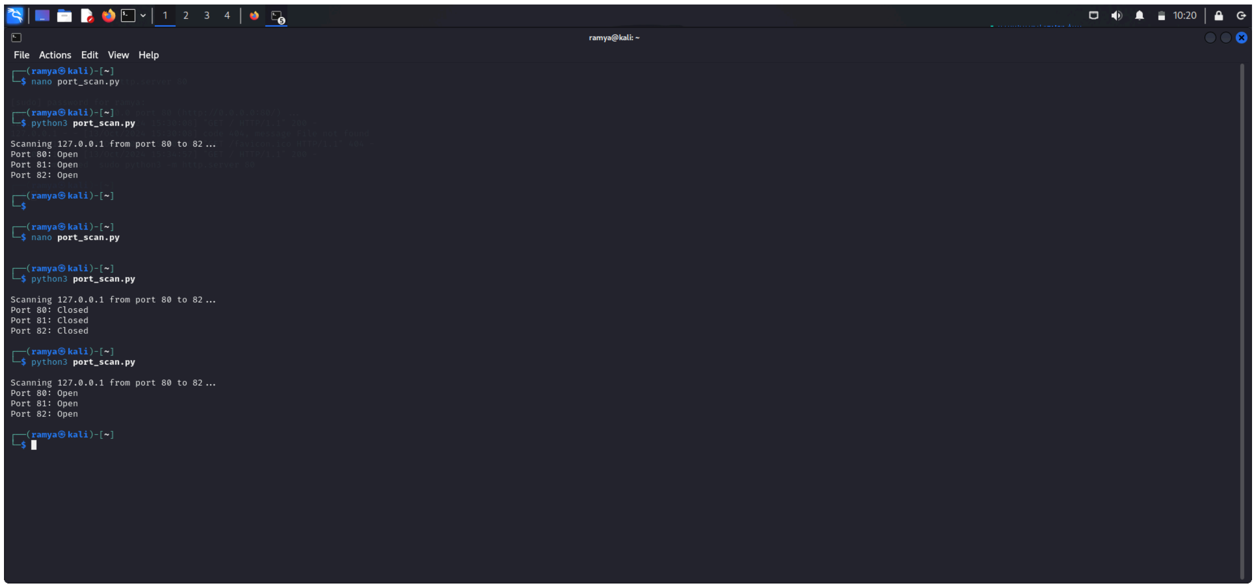
```
┌──(ramya㉿kali)-[~]
└─$ nc -l -p 81


┌──(ramya㉿kali)-[~]
└─$ nc -l -p 81

█
```



```
        -s addr                 local source address
        -T tos                  set Type Of Service
        -t                      answer TELNET negotiation
        -u                      UDP mode
        -v                      verbose [use twice to be more verbose]
        -w secs                 timeout for connects and final net reads
        -C                      Send CRLF as line-ending
        -z                      zero-I/O mode [used for scanning]
port numbers can be individual or ranges: lo-hi [inclusive];
hyphens in port names must be backslash escaped (e.g. 'ftp\-data').

┌──(ramya㉿kali)-[~]
└─$ sudo nc -l -p 80

Can't grab 0.0.0.0:80 with bind

┌──(ramya㉿kali)-[~]
└─$ sudo netstat -tuln | grep :80

tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN

┌──(ramya㉿kali)-[~]
└─$ sudo kill <PID>

zsh: parse error near `\n'

┌──(ramya㉿kali)-[~]
└─$ sudo kill 1234


┌──(ramya㉿kali)-[~]
└─$ sudo netstat -tulnp | grep :80

tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      11375/python3

┌──(ramya㉿kali)-[~]
└─$ sudo kill 11375

┌──(ramya㉿kali)-[~]
└─$ sudo netstat -tulnp | grep :80


┌──(ramya㉿kali)-[~]
└─$ sudo nc -l -p 80


┌──(ramya㉿kali)-[~]
└─$ █
```



```
┌──(ramya㉿kali)-[~]
└─$ nc -l -p 82

┌──(ramya㉿kali)-[~]
└─$ nc -l -p 82
█
```



```
┌──(ramya㉿kali)-[~]
└─$ nano port_scan.py

┌──(ramya㉿kali)-[~]
└─$ python3 port_scan.py
Scanning 127.0.0.1 from port 80 to 82...
Port 80: Open
Port 81: Open
Port 82: Open

┌──(ramya㉿kali)-[~]
└─$

┌──(ramya㉿kali)-[~]
└─$ nano port_scan.py

┌──(ramya㉿kali)-[~]
└─$ python3 port_scan.py
Scanning 127.0.0.1 from port 80 to 82...
Port 80: Closed
Port 81: Closed
Port 82: Closed

┌──(ramya㉿kali)-[~]
└─$ python3 port_scan.py
Scanning 127.0.0.1 from port 80 to 82...
Port 80: Open
Port 81: Open
Port 82: Open

┌──(ramya㉿kali)-[~]
└─$ █
```

at first the ports are closed there are no services are running in the ports i just tried to open the ports and run the python port code again

What is the purpose of port scanning in network security?

In network security, port scanning is used to find open ports on a system that is networked. Every open port may be a sign that a particular service (such as SSH or a web server) is using that port. Knowing these ports is beneficial:

- Evaluate security vulnerabilities: If the service using the port has security issues, open ports may serve as entry points for attackers.
- Examine network services: ensuring that just the services that are required are operating.
- Investigate network problems by confirming that specific services are operational and reachable.

What is the difference between an open port and a closed port? (can you provide a screenshot from your results)

A service (such as a web server or SSH) that is actively seeking connections is said to have an open port. Other devices connected to the network can access it.

- Closed Port: This port is not being used by any services. Nothing is accepting connections on that port, despite the fact that the system can be reached.

In the context of this program, what does `socket.connect_ex()` do, and how is it used to determine if a port is open or closed? (can you provide a screenshot from your results)

`socket.connect_ex()`: It attempts to make a connection to a specific port on the target IP.

- If the port is open, it returns `0`, meaning the connection was successful.
- If the port is closed or unreachable, it returns a non-zero error code.

What role does the `sock.settimeout(1)` function play in this program? Why is setting a timeout important? (can you provide a screenshot from your results)

Sock.settimeout(1): This limits the amount of time the script will wait for a response while attempting to connect to a port to one second. It stops waiting and considers the port closed if the connection takes longer than this.

What makes this significant?

makes sure the script doesn't hang: The application might wait endlessly for a response if there was no timeout specified, which would slow down the scan.

speeds up scanning: The application may immediately ascertain whether a port is closed or unresponsive without having to wait a long time if an appropriate timeout is set.