Bakka Ramyasree

UDP Port Scanner

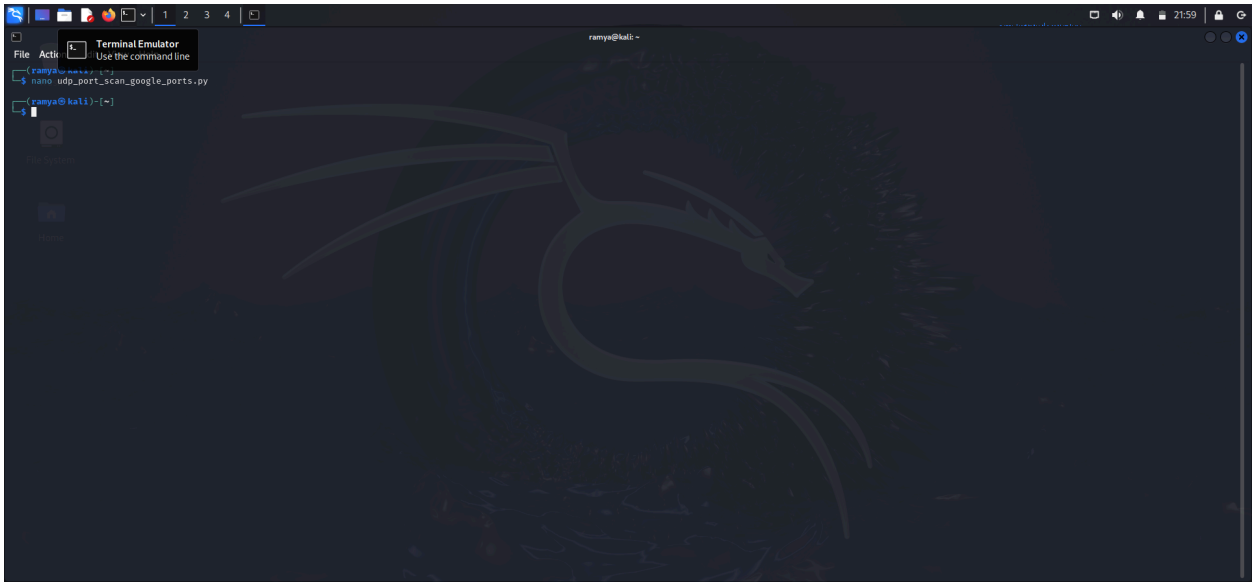
Why do UDP ports often show no response when scanned, and how does this differ from TCP port scanning? (change code to show in result)

UDP is a connectionless protocol, meaning it does not establish a connection before sending data. Many UDP services might not respond to unexpected or unsolicited packets, leading to the common observation of "no response" when scanned. Unlike TCP, which typically responds with SYN/ACK or RST packets for open or closed ports, UDP may simply not reply, making it harder to determine port states.

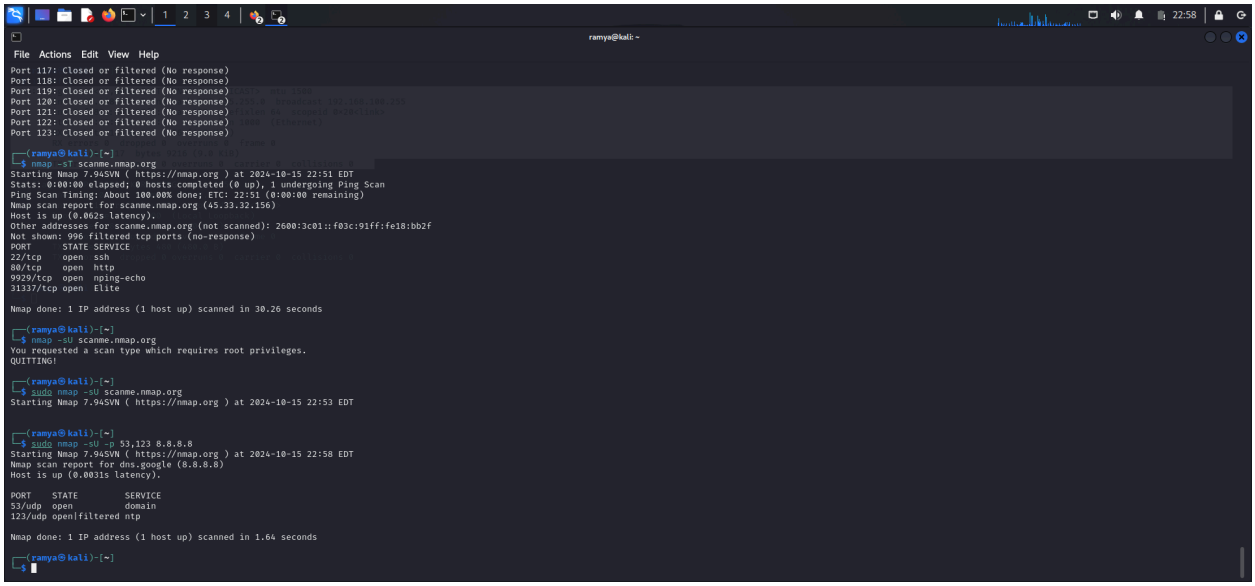
What does it mean when the scanner reports "Open or filtered" for a UDP port? (show in result)

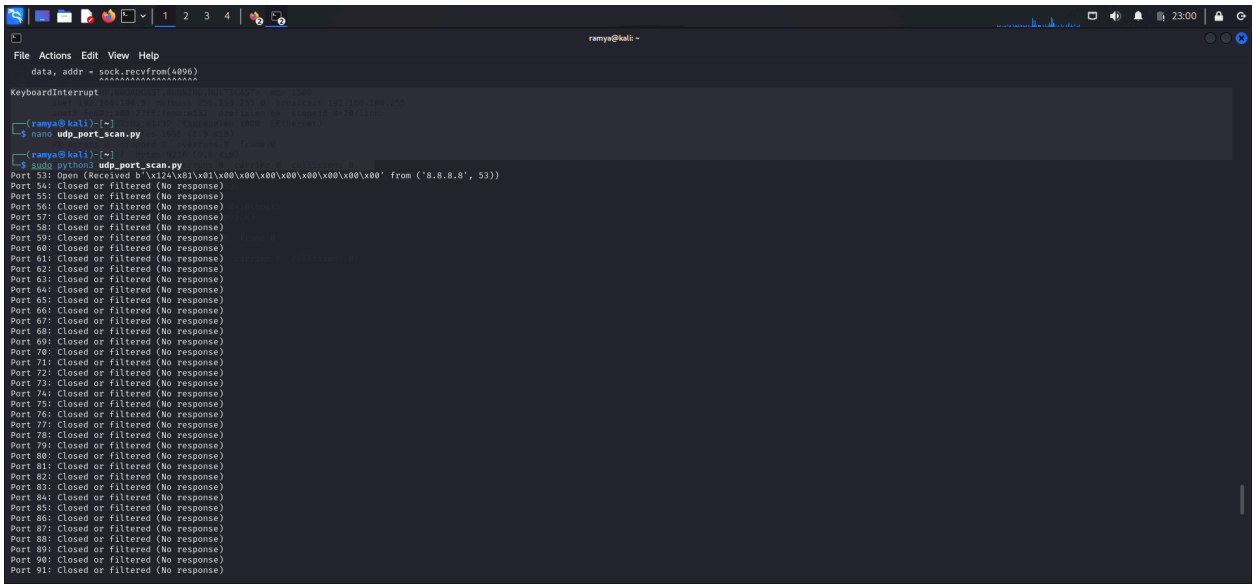
When the scanner reports "Open or filtered," it means that:

- Open: The port is accepting packets, and the service is running.
- Filtered: The port is either closed or a firewall is blocking the packet, preventing a response. This makes it difficult to determine the exact state.

- 
1 2 3 4
21:59
ramya@kali: ~
File Acti
Terminal Emulator
Use the command line
$ nano udp_port_scan_google_ports.py
(ramya㉿kali)-[~]
$


1 2 3 4
22:00
ramya@kali: ~
File Actions Edit View Help
GNU nano 8.0
udp_port_scan_google_ports.py
import socket
def udp_port_scan(ip, start_port, end_port):
print(f"Scanning {ip} for open UDP ports from {start_port} to {end_port} ...")
for port in range(start_port, end_port + 1):
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.settimeout(1)
try:
sock.sendto(b"", (ip, port))
data, addr = sock.recvfrom(1024) # Trying to receive a response
print(f"Port {port}: Open")
except socket.timeout:
print(f"Port {port}: Closed or filtered")
finally:
sock.close()
# Using Google DNS (8.8.8.8) as the target IP and specific ports for DNS (53) and NTP (123)
target_ip = '8.8.8.8'
start_port = 53
end_port = 123
# Perform UDP port scan
udp_port_scan(target_ip, start_port, end_port)
[Read 23 lines]
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark M-] To Bracket M-B Previous ^B Back ^Q Prev Word ^A Home
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo M-6 Copy ^_ Where Was M-F Next ^F Forward ^W Next Word ^E End





How does the use of `socket.timeout()` help in the UDP scanning process? (show in result)

- The `socket.timeout()` method allows the program to avoid hanging indefinitely when waiting for a response. If no response is received within the specified time frame, the program can move on to the next port, thus making the scanning process more efficient.

What would happen if no timeout were set during the UDP scan? How would this affect the overall scanning process? (show in result by changing code)

Without a timeout, the `recvfrom()` call will block indefinitely if no response is received. This would result in the program hanging on any port that does not reply, making it appear unresponsive and prolonging the scanning process significantly.

Modified the code by removing socket.timeout()

The output shows **Blocking on Non-Responsive Ports**: When you run this modified code, if a UDP port does not respond, your script will hang indefinitely on that port. You won't receive a "Closed or filtered" message; instead, the script will simply wait until a response is received or an error occurs

```
GNU nano 8.2                                udp_port_scan_google_ports.py
import socket

def udp_port_scan(ip, start_port, end_port):
    print(f"Scanning {ip} for open UDP ports from {start_port} to {end_port} ...")
    for port in range(start_port, end_port + 1):
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        try:
            sock.sendto(b"", (ip, port))
            data, addr = sock.recvfrom(1024)  # Trying to receive a response
            print(f"Port {port}: Open")
        except socket.timeout:
            print(f"Port {port}: Closed or filtered")
        finally:
            sock.close()

# Using Google DNS (8.8.8.8) as the target IP and specific ports for DNS (53) and NTP (123)
target_ip = '8.8.8.8'
start_port = 53
end_port = 123

# Perform UDP port scan
udp_port_scan(target_ip, start_port, end_port)

                                   [ Read 22 lines ]
^G Help      ^O Write Out   ^F Where Is    ^K Cut       ^T Execute   ^C Location  M-U Undo      M-A Set Mark   M-] To Bracket  M-B Previous   ^← Back        ^▭ Prev Word   ^A Home
^X Exit      ^R Read File   ^\ Replace     ^U Paste     ^J Justify   ^/ Go To Line M-E Redo      M-6 Copy       ^B Where Was    M-F Next       ^→ Forward     ^▭ Next Word   ^E End
```

```
┌──(ramya㉿kali)-[~]
└─$ python3  udp_port_scan_google_ports.py
Scanning 8.8.8.8 for open UDP ports from 53 to 123 ...
Port 53: Closed or filtered
Port 54: Closed or filtered
Port 55: Closed or filtered
Port 56: Closed or filtered
Port 57: Closed or filtered
Port 58: Closed or filtered
Port 59: Closed or filtered
Port 60: Closed or filtered
Port 61: Closed or filtered
Port 62: Closed or filtered
Port 63: Closed or filtered
Port 64: Closed or filtered
Port 65: Closed or filtered
Port 66: Closed or filtered
Port 67: Closed or filtered
Port 68: Closed or filtered
Port 69: Closed or filtered
Port 70: Closed or filtered
Port 71: Closed or filtered
Port 72: Closed or filtered
Port 73: Closed or filtered
Port 74: Closed or filtered
Port 75: Closed or filtered
Port 76: Closed or filtered
Port 77: Closed or filtered
Port 78: Closed or filtered
Port 79: Closed or filtered
Port 80: Closed or filtered
^CTraceback (most recent call last):
  File "/home/ramya/udp_port_scan_google_ports.py", line 23, in <module>
    udp_port_scan(target_ip, start_port, end_port)
  File "/home/ramya/udp_port_scan_google_ports.py", line 10, in udp_port_scan
    data, addr = sock.recvfrom(1024)  # Trying to receive a response
                 ^^^^^^^^^^^^^^^^^^
KeyboardInterrupt

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

┌──(ramya㉿kali)-[~]
└─$ python3  udp_port_scan_google_ports.py
Scanning 8.8.8.8 for open UDP ports from 53 to 123 ...
```