Network Diagnostics and System Analysis Project

Project Overview

This project demonstrates practical IT troubleshooting and network analysis skills using Linux-based tools. The objective is to identify network interfaces, discover active hosts, analyze open ports, and visualize network topology — key tasks relevant to IT support environments. All work is performed in a controlled virtual environment.

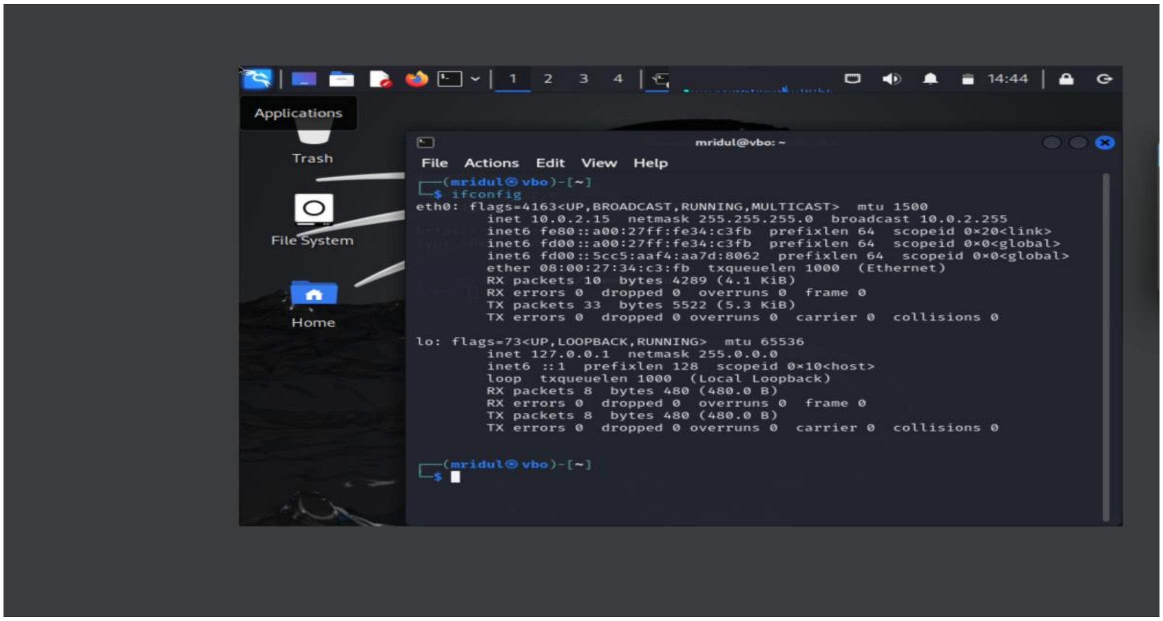
Tools and Technologies Used

- Kali Linux
- Netdiscover
- Nmap / Zenmap (GUI)
- Traceroute
- Dig (DNS Lookup)
- ARP and ifconfig utilities
- Wireshark (optional for packet analysis)

Methodology

1. Network Interface Configuration

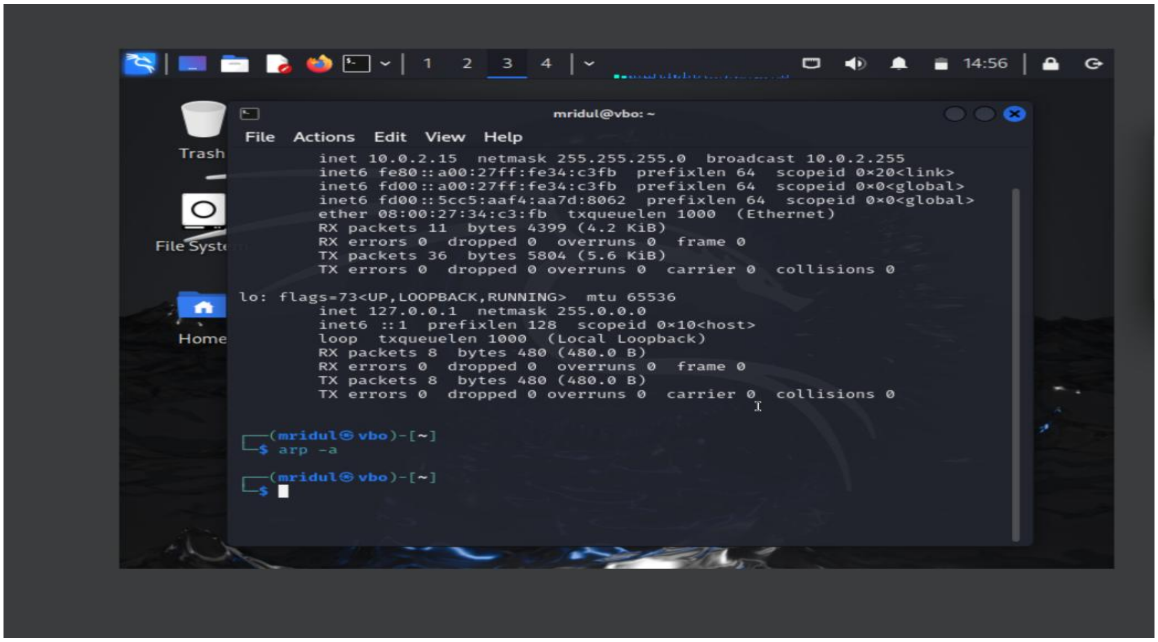
Used the `ifconfig` command to display network interfaces and confirm active IP configuration. Verified that the system is connected and able to communicate within the local network.



```
┌──(mridul㉿vbo)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe34:c3fb  prefixlen 64  scopeid 0x20<link>
        inet6 fd00::a00:27ff:fe34:c3fb  prefixlen 64  scopeid 0x0<global>
        inet6 fd00::5cc5:aaf4:aa7d:8062  prefixlen 64  scopeid 0x0<global>
        ether 08:00:27:34:c3:fb  txqueuelen 1000  (Ethernet)
        RX packets 10  bytes 4289 (4.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 33  bytes 5522 (5.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 480 (480.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 480 (480.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

┌──(mridul㉿vbo)-[~]
└─$ 
```

2. ARP Cache Inspection

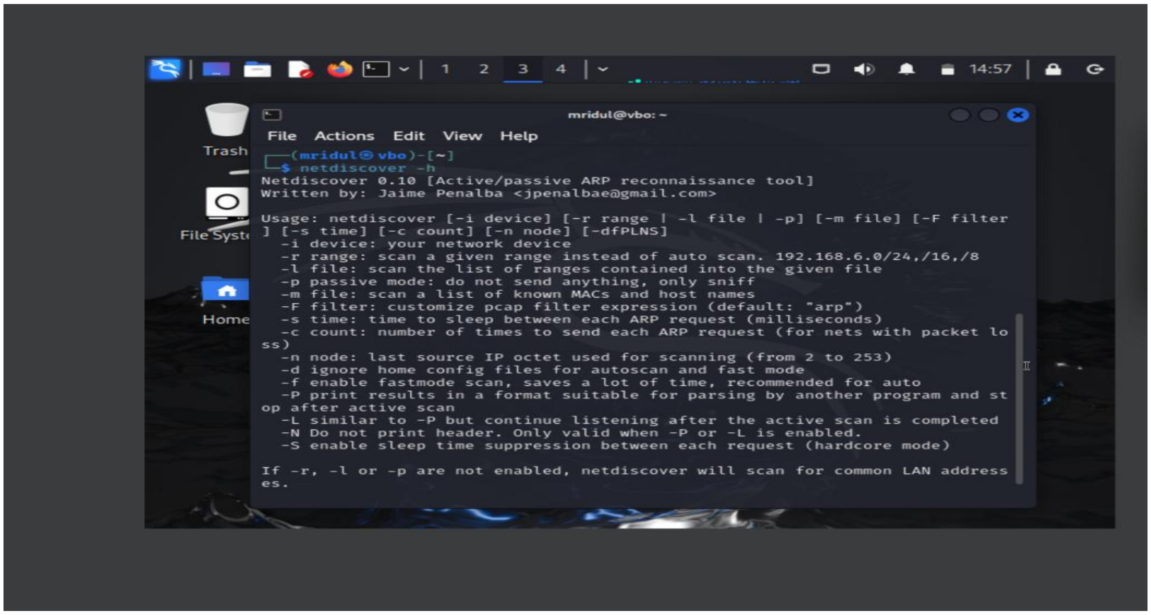
Executed `arp -a` to view the Address Resolution Protocol cache, identifying IP and MAC address relationships. This step helps detect active devices and confirm ARP table functionality.



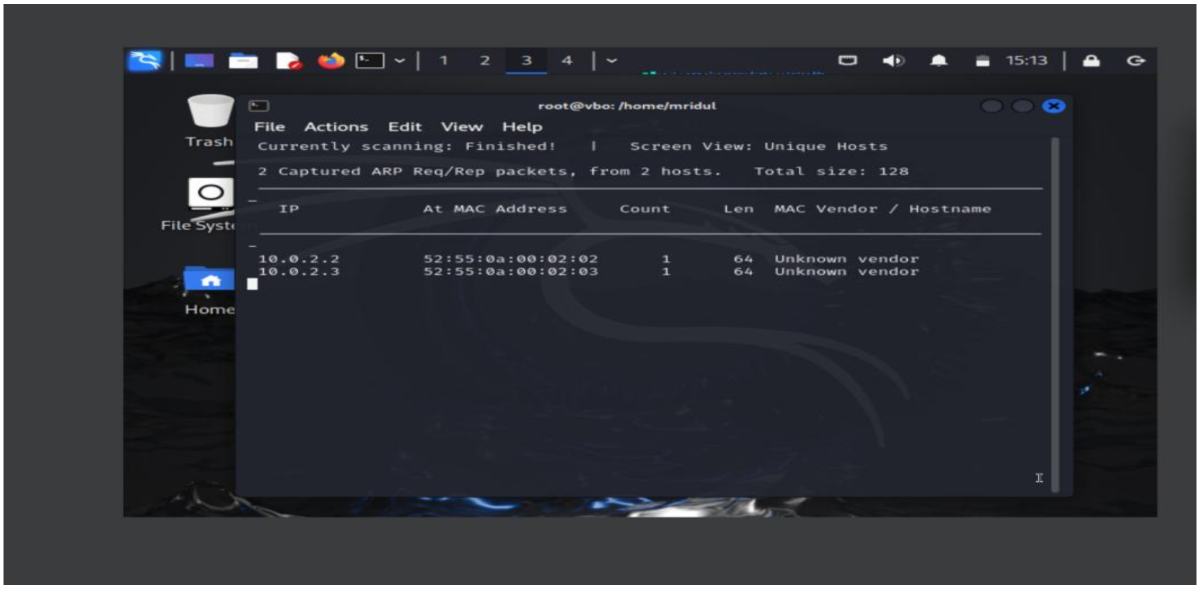
```
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe34:c3fb  prefixlen 64  scopeid 0x20<link>
        inet6 fd00::a00:27ff:fe34:c3fb  prefixlen 64  scopeid 0x0<global>
        inet6 fd00::5cc5:aaf4:aa7d:8062  prefixlen 64  scopeid 0x0<global>
        ether 08:00:27:34:c3:fb  txqueuelen 1000  (Ethernet)
        RX packets 11  bytes 4399 (4.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 36  bytes 5804 (5.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 480 (480.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 480 (480.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


┌──(mridul㉿vbo)-[~]
└─$ arp -a

┌──(mridul㉿vbo)-[~]
└─$ 
```

3. Network Discovery using Netdiscover

Ran the `netdiscover -i eth0 -r [subnet range]` command to identify live hosts within the subnet. This revealed the IP and MAC addresses of connected devices in the local network.



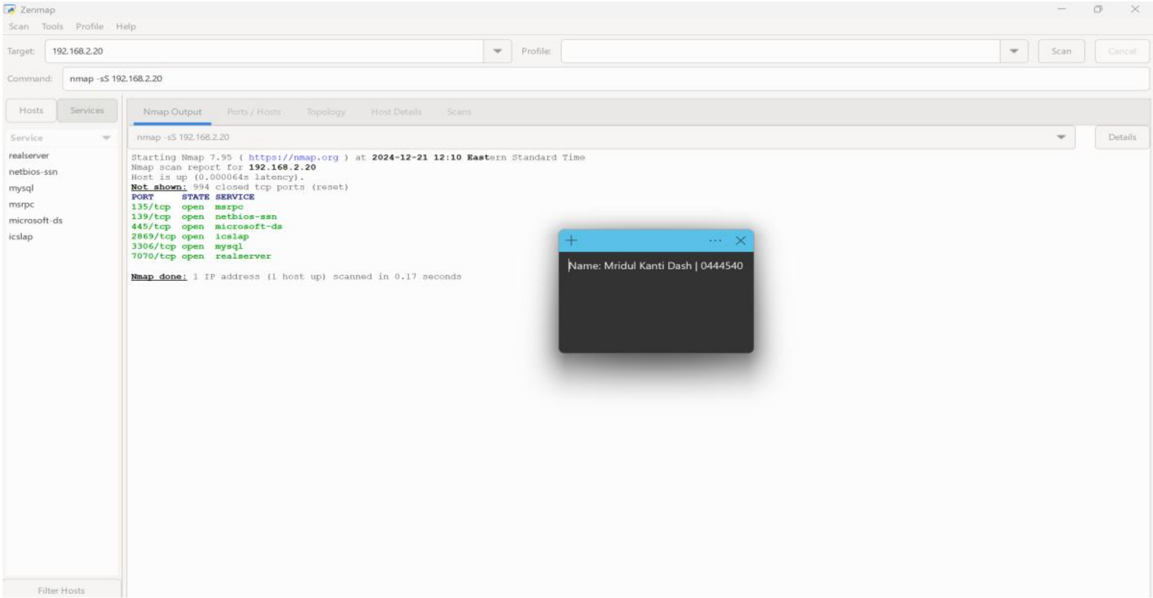
```
┌──(mridul㉿vbo)-[~]
└─$ netdiscover -h
Netdiscover 0.10 [Active/passive ARP reconnaissance tool]
Written by: Jaime Penalba <jpenalbae@gmail.com>

Usage: netdiscover [-i device] [-r range | -l file | -p] [-m file] [-F filter
] [-s time] [-c count] [-n node] [-dfPLNS]
  -i device: your network device
  -r range: scan a given range instead of auto scan. 192.168.6.0/24,/16,/8
  -l file: scan the list of ranges contained into the given file
  -p passive mode: do not send anything, only sniff
  -m file: scan a list of known MACs and host names
  -F filter: customize pcap filter expression (default: "arp")
  -s time: time to sleep between each ARP request (milliseconds)
  -c count: number of times to send each ARP request (for nets with packet lo
ss)
  -n node: last source IP octet used for scanning (from 2 to 253)
  -d ignore home config files for autoscan and fast mode
  -f enable fastmode scan, saves a lot of time, recommended for auto
  -P print results in a format suitable for parsing by another program and st
op after active scan
  -L similar to -P but continue listening after the active scan is completed
  -N Do not print header. Only valid when -P or -L is enabled.
  -S enable sleep time suppression between each request (hardcore mode)

If -r, -l or -p are not enabled, netdiscover will scan for common LAN address
es.
```



```
Currently scanning: Finished!   |   Screen View: Unique Hosts

2 Captured ARP Req/Rep packets, from 2 hosts.   Total size: 128
_____________________________________________________________________________
   IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
10.0.2.2        52:55:0a:00:02:02      1       64  Unknown vendor
10.0.2.3        52:55:0a:00:02:03      1       64  Unknown vendor
```

4. Port Scanning using Nmap

Performed SYN and intensive TCP/UDP scans using Nmap to detect open or closed ports on identified hosts. Zenmap (the GUI version) was also used to visualize host topology and generate scan reports.



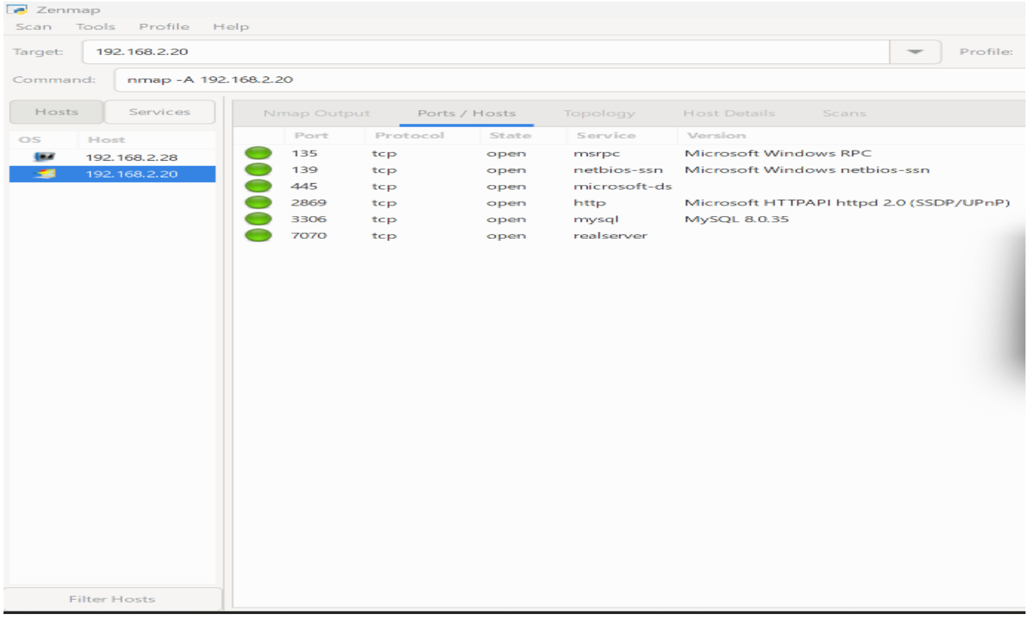
Target: 192.168.2.20

Command: nmap -sS 192.168.2.20

realserver
netbios-ssn
mysql
msrpc
microsoft-ds
icslap

```
nmap -sS 192.168.2.20

Starting Nmap 7.95 ( https://nmap.org ) at 2024-12-21 12:10 Eastern Standard Time
Nmap scan report for 192.168.2.20
Host is up (0.000064s latency).
Not shown: 994 closed tcp ports (reset)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
2869/tcp open  icslap
3306/tcp open  mysql
7070/tcp open  realserver

Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds
```

Name: Mridul Kanti Dash | 0444540



Target: 192.168.2.20

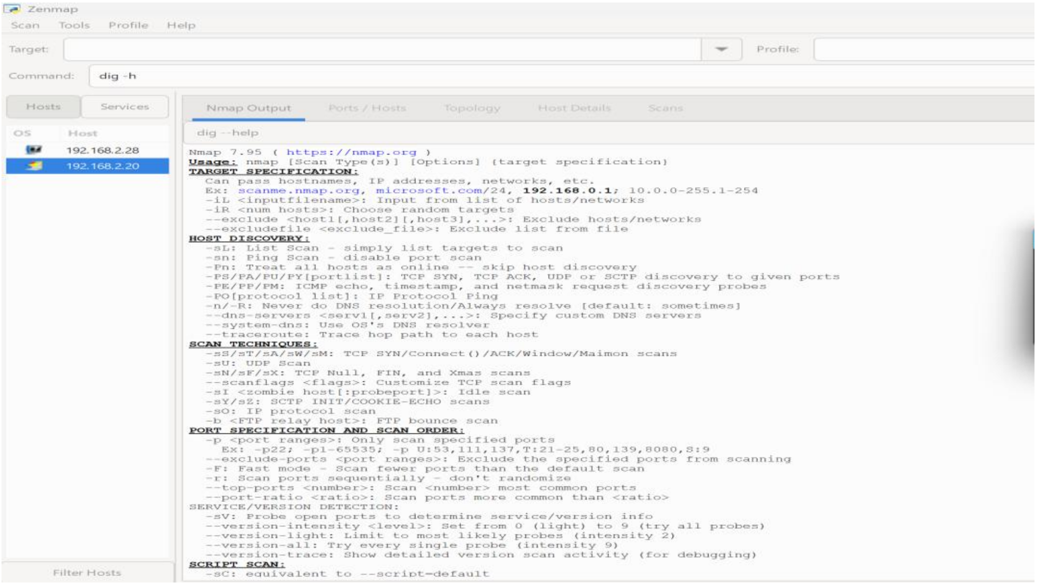
Command: nmap -A 192.168.2.20

OS	Host
	192.168.2.28
	192.168.2.20

	Port	Protocol	State	Service	Version
●	135	tcp	open	msrpc	Microsoft Windows RPC
●	139	tcp	open	netbios-ssn	Microsoft Windows netbios-ssn
●	445	tcp	open	microsoft-ds	
●	2869	tcp	open	http	Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
●	3306	tcp	open	mysql	MySQL 8.0.35
●	7070	tcp	open	realserver	

5. DNS Analysis using Dig

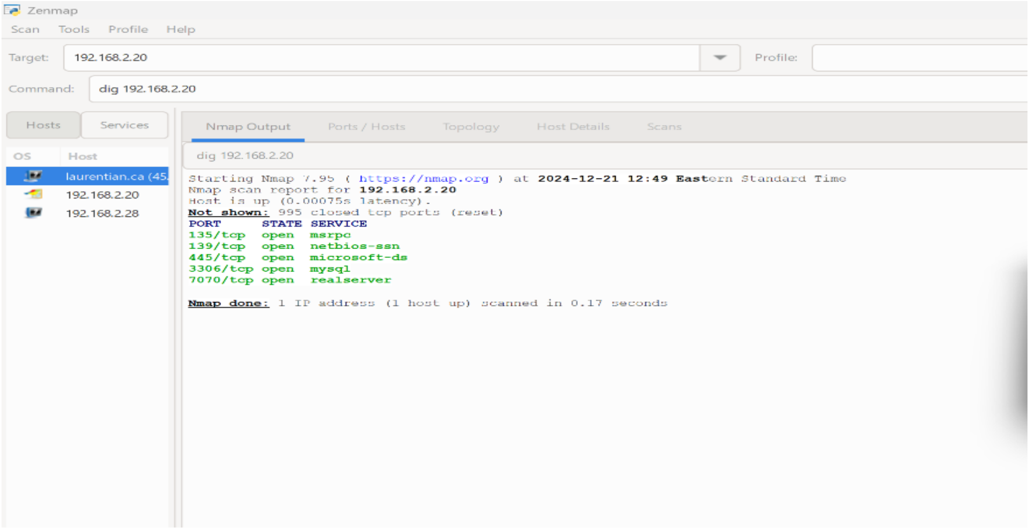
Utilized the `dig` command to perform DNS lookups, zone transfers, and other network name resolution diagnostics. This test helps confirm domain-related configurations and troubleshooting steps.



Command: dig -h

OS	Host
	192.168.2.28
	192.168.2.20

```
dig --help
Nmap 7.95 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN, TCP ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
  -p <port ranges>: Only scan specified ports
    Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
  --exclude-ports <port ranges>: Exclude the specified ports from scanning
  -F: Fast mode - Scan fewer ports than the default scan
  -r: Scan ports sequentially - don't randomize
  --top-ports <number>: Scan <number> most common ports
  --port-ratio <ratio>: Scan ports more common than <ratio>
SERVICE/VERSION DETECTION:
  -sV: Probe open ports to determine service/version info
  --version-intensity <level>: Set from 0 (light) to 9 (try all probes)
  --version-light: Limit to most likely probes (intensity 2)
  --version-all: Try every single probe (intensity 9)
  --version-trace: Show detailed version scan activity (for debugging)
SCRIPT SCAN:
  -sC: equivalent to --script=default
```



Target: 192.168.2.20

Profile:

Command: dig 192.168.2.20

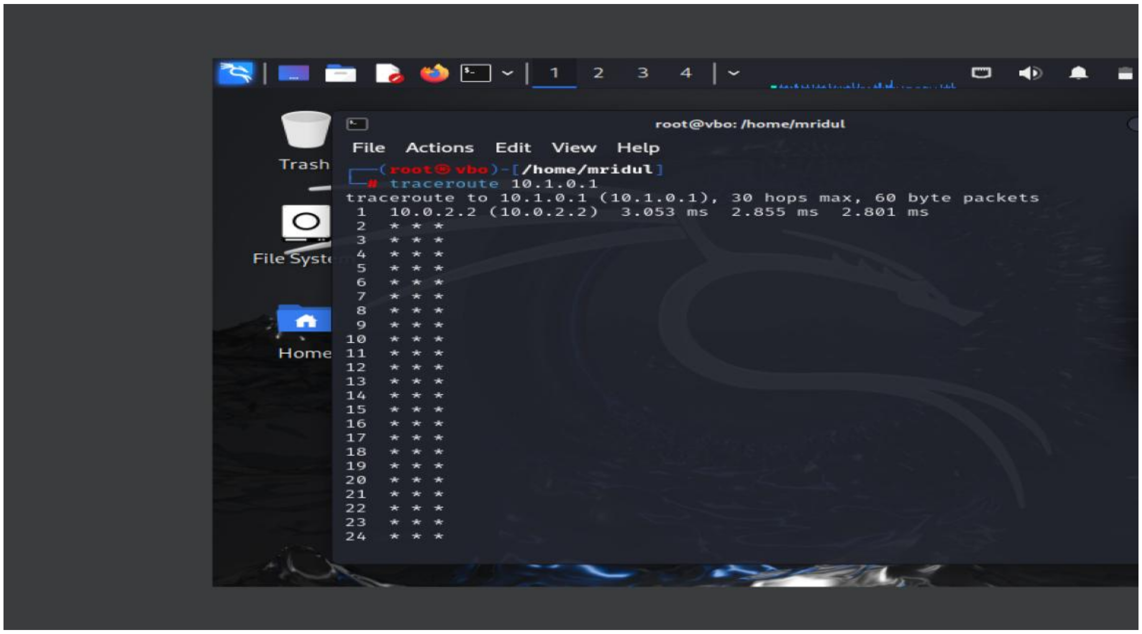
laurentian.ca (45.
192.168.2.20
192.168.2.28

dig 192.168.2.20

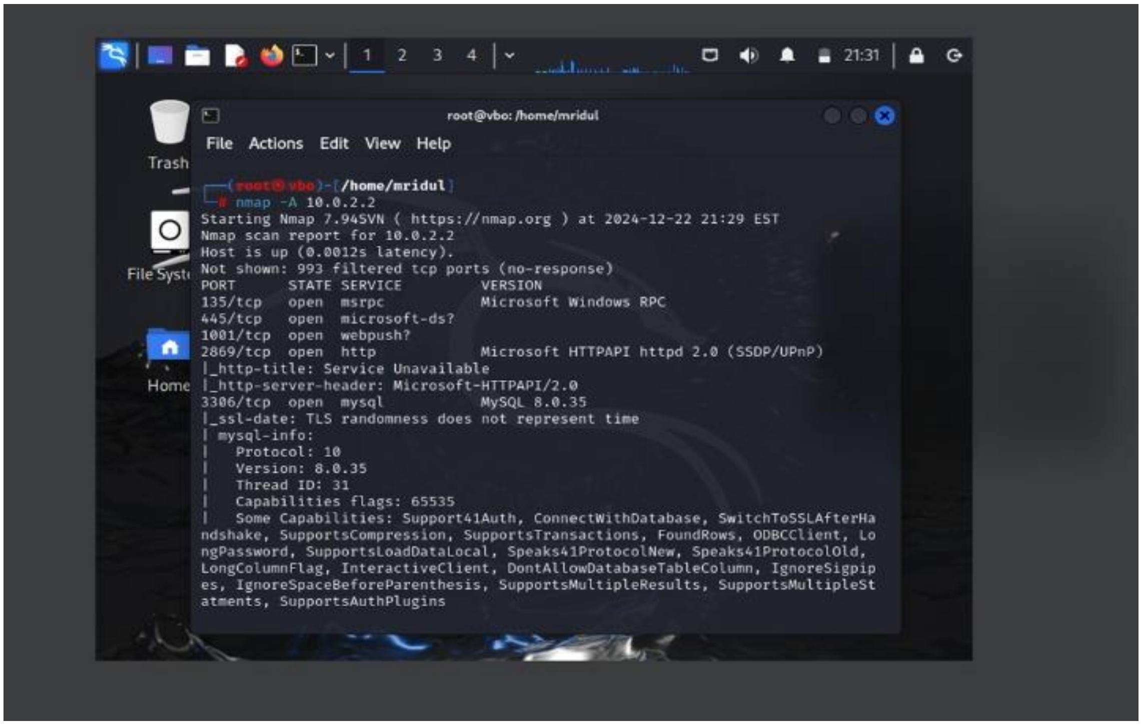
```
Starting Nmap 7.95 ( https://nmap.org ) at 2024-12-21 12:49 Eastern Standard Time
Nmap scan report for 192.168.2.20
Host is up (0.00075s latency).
Not shown: 995 closed tcp ports (reset)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3306/tcp open  mysql
7070/tcp open  realserver

Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds
```

6. Traceroute Network Mapping

Ran `traceroute [destination IP or domain]` to map the path and hops data packets take to reach the target host. This is used to analyze routing, latency, and potential network bottlenecks.



```
┌──(root㉿vbo)-[/home/mridul]
└─# traceroute 10.1.0.1
traceroute to 10.1.0.1 (10.1.0.1), 30 hops max, 60 byte packets
 1  10.0.2.2 (10.0.2.2)  3.053 ms  2.855 ms  2.801 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
```

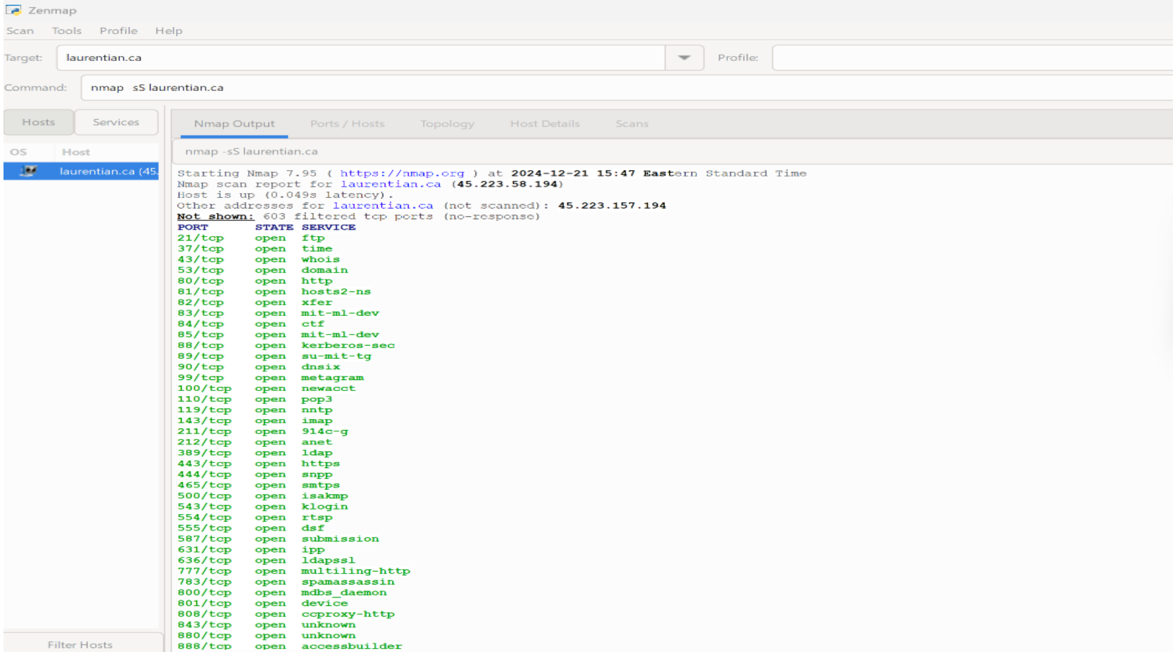


```
┌──(root㉿vbo)-[/home/mridul]
└─# nmap -A 10.0.2.2
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-22 21:29 EST
Nmap scan report for 10.0.2.2
Host is up (0.0012s latency).
Not shown: 993 filtered tcp ports (no-response)
PORT      STATE SERVICE         VERSION
135/tcp   open  msrpc           Microsoft Windows RPC
445/tcp   open  microsoft-ds?
1001/tcp  open  webpush?
2869/tcp  open  http            Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Service Unavailable
|_http-server-header: Microsoft-HTTPAPI/2.0
3306/tcp  open  mysql           MySQL 8.0.35
|_ssl-date: TLS randomness does not represent time
| mysql-info:
|   Protocol: 10
|   Version: 8.0.35
|   Thread ID: 31
|   Capabilities flags: 65535
|   Some Capabilities: Support41Auth, ConnectWithDatabase, SwitchToSSLAfterHa
ndshake, SupportsCompression, SupportsTransactions, FoundRows, ODBCClient, Lo
ngPassword, SupportsLoadDataLocal, Speaks41ProtocolNew, Speaks41ProtocolOld,
LongColumnFlag, InteractiveClient, DontAllowDatabaseTableColumn, IgnoreSigpip
es, IgnoreSpaceBeforeParenthesis, SupportsMultipleResults, SupportsMultipleSt
atments, SupportsAuthPlugins
```

7. Network Topology Visualization

Used Zenmap to generate topology diagrams showing host relationships and network paths. This visualization helps IT professionals understand system connectivity and troubleshoot more effectively.

Intensive scan of tcp and udp : For this I have used our university domain address



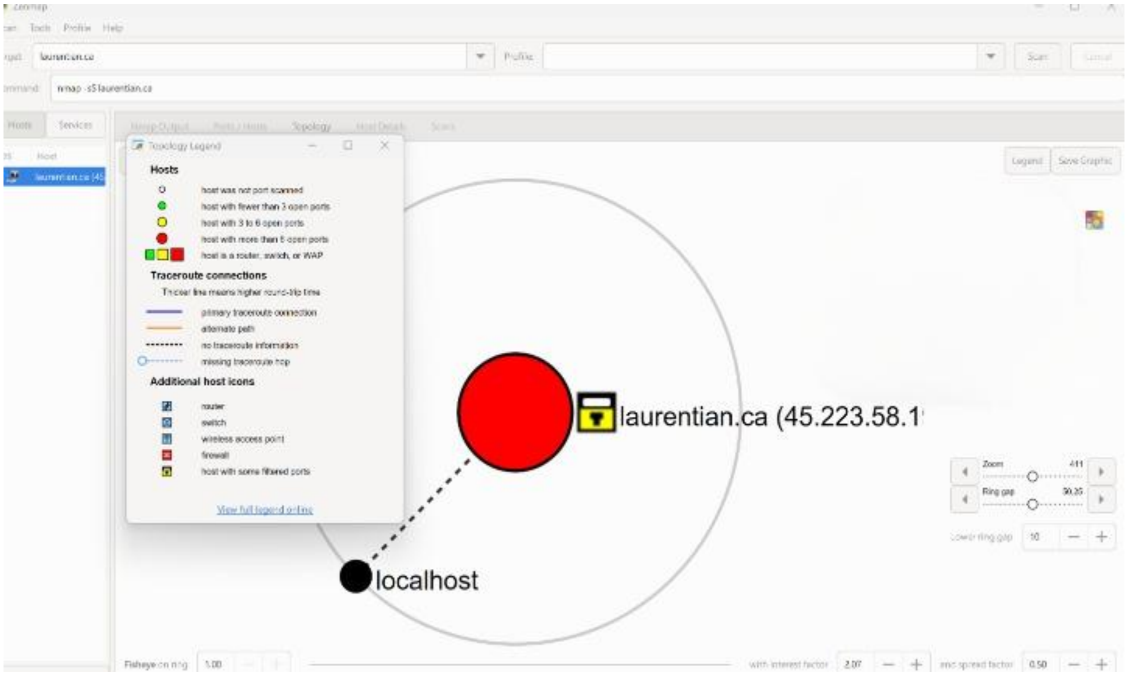
Target: laurentian.ca

Profile:

Command: nmap -sS laurentian.ca

```
nmap -sS laurentian.ca
```

```
Starting Nmap 7.95 ( https://nmap.org ) at 2024-12-21 15:47 Eastern Standard Time
Nmap scan report for laurentian.ca (45.223.58.194)
Host is up (0.049s latency).
Other addresses for laurentian.ca (not scanned): 45.223.157.194
Not shown: 603 filtered tcp ports (no-response)
PORT      STATE SERVICE
21/tcp    open  ftp
37/tcp    open  time
43/tcp    open  whois
53/tcp    open  domain
80/tcp    open  http
81/tcp    open  hosts2-ns
82/tcp    open  xfer
83/tcp    open  mit-ml-dev
84/tcp    open  ctf
85/tcp    open  mit-ml-dev
88/tcp    open  kerberos-sec
89/tcp    open  su-mit-tg
90/tcp    open  dnsix
99/tcp    open  metagram
100/tcp   open  newacct
110/tcp   open  pop3
119/tcp   open  nntp
143/tcp   open  imap
211/tcp   open  914c-g
212/tcp   open  anet
389/tcp   open  ldap
443/tcp   open  https
444/tcp   open  snpp
465/tcp   open  smtps
500/tcp   open  isakmp
543/tcp   open  klogin
554/tcp   open  rtsp
555/tcp   open  dsf
587/tcp   open  submission
631/tcp   open  ipp
636/tcp   open  ldapssl
777/tcp   open  multiling-http
783/tcp   open  spamassassin
800/tcp   open  mdbs_daemon
801/tcp   open  device
808/tcp   open  ccproxy-http
843/tcp   open  unknown
880/tcp   open  unknown
888/tcp   open  accessbuilder
```



```
nmap -sS laurentian.ca
```

Results and Observations

The network diagnostic tests successfully identified active hosts, open service ports, and routing paths. DNS queries provided insights into domain-level resolution, while traceroute results confirmed connectivity and hop efficiency. The Nmap and Zenmap scans offered a detailed overview of the network structure, supporting IT troubleshooting and system performance validation.

Conclusion

This project provided hands-on experience with network diagnostic and analysis tools used in IT support. It strengthened understanding of host discovery, DNS resolution, and connectivity analysis. These skills directly translate to real-world helpdesk and system administration scenarios, where identifying network issues and documenting system configurations are essential tasks.