Linux Network Troubleshooting

Linux Networking

Serhii Zakharchenko

TRAINING CENTER

Agenda

- General troubleshooting procedures
- Linux monitoring and troubleshooting tools

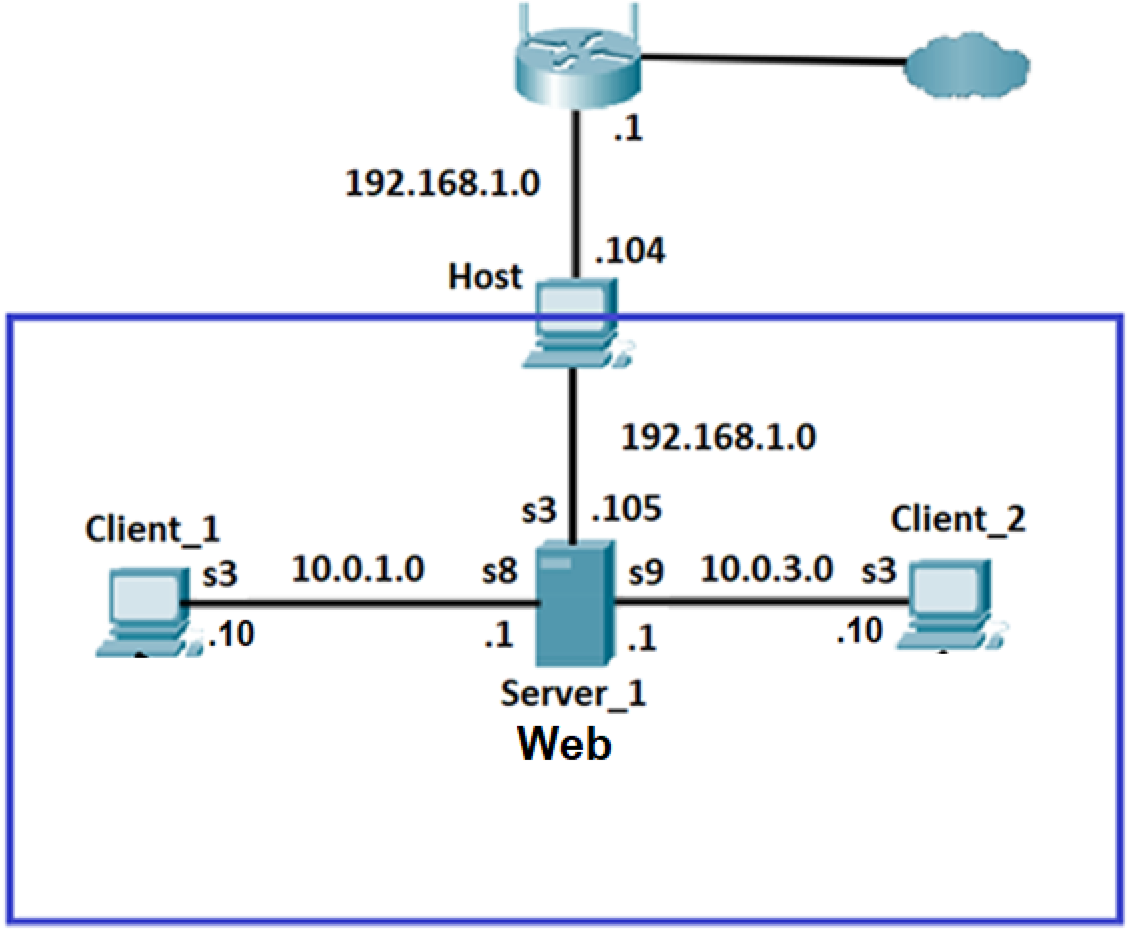
General troubleshooting procedures

Documenting the Network

- Network documentation refers to the practice of capturing and maintaining records related to the network and the devices connected to it.
- This documentation can encompass a range of specialized and technical information that’s included according to its relevance.
- The documentation that’s maintained should give administrators insight into the network and how it performs.
- Comprehensive network documentation is of crucial importance, because it supports troubleshooting activities and can help more quickly identify the root cause of any network issues.
- This documentation includes:
    - Network topology map
    - Configuration files, including network configuration files and end-system configuration files
    - A baseline performance level

Network topology map

- A network topology map provides a visual representation of how all of your network parts are related and connected.
- The map should include each network segment, the routers that connect them, and the gateways, servers, and hardware associated with each.
- Server details should include the server name, the role performed by the server, and the IP address.
- Once captured in a visual map, these up-to-date details will provide an intuitive way for viewers to troubleshoot issues with minimal downtime.



End-system Configuration Table

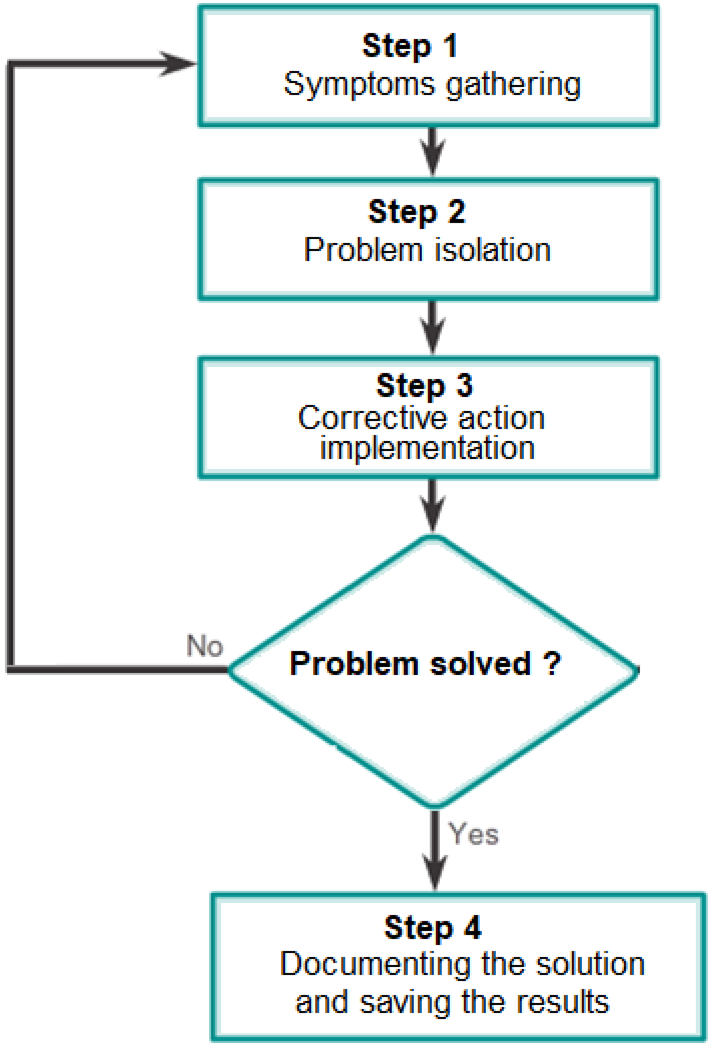
End-system configuration files focus on the hardware and software used in end-system devices, such as servers, network management consoles, and user workstations. End-system configuration table includes:

- Device name (purpose);
- Operating system and version;
- IP addresses, Subnet mask and prefix length;
- Default gateway, DNS server.

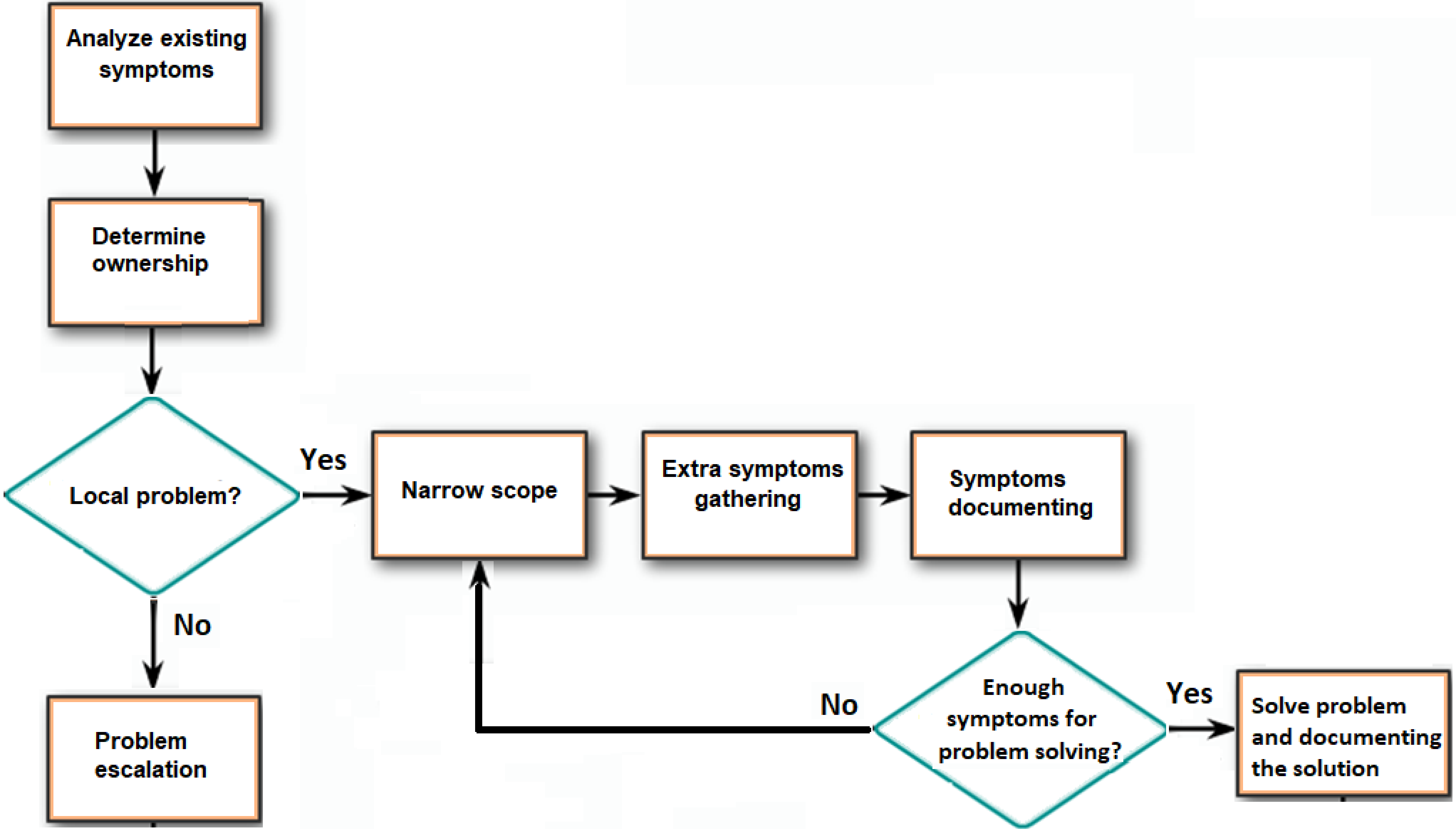
Device Name, Purpose	Operating System	Interface	MAC Address	IP Address	Default Gateway
Client_1, Workstation	Ubuntu 20.04	s3	0C50.0FDB.8E77	10.0.1.10/24	10.0.1.1
Client_2, Workstation	CentOS 8	s3	0CD0.FFB7.A43A	10.0.3.10/24	10.0.3.1
Server_1, Web	Ubuntu 22.04	s3	0C90.2B85.BD3C	192.168.1.105/24	192.168.1.1
		s8	0C06.2ADB.C886	10.0.1.1	
		s9	0C01.644E.9157	10.0.3.1	

General Troubleshooting Procedures

- **Step 1. Symptoms gathering.** Troubleshooting begins with gathering and documenting symptoms from the network, end systems, and users.
- **Step 2. Problem isolation** is the process of eliminating variables until a single problem, or a set of related problems has been identified as the cause.
- **Step 3. Corrective action implementation.** Having identified the cause of the problem, the network administrator works to correct the problem by implementing, testing, and documenting possible solutions.
- **Step 4. Documenting the solution and saving the results.** The administrator carefully documents the problem and the ways to solve it.

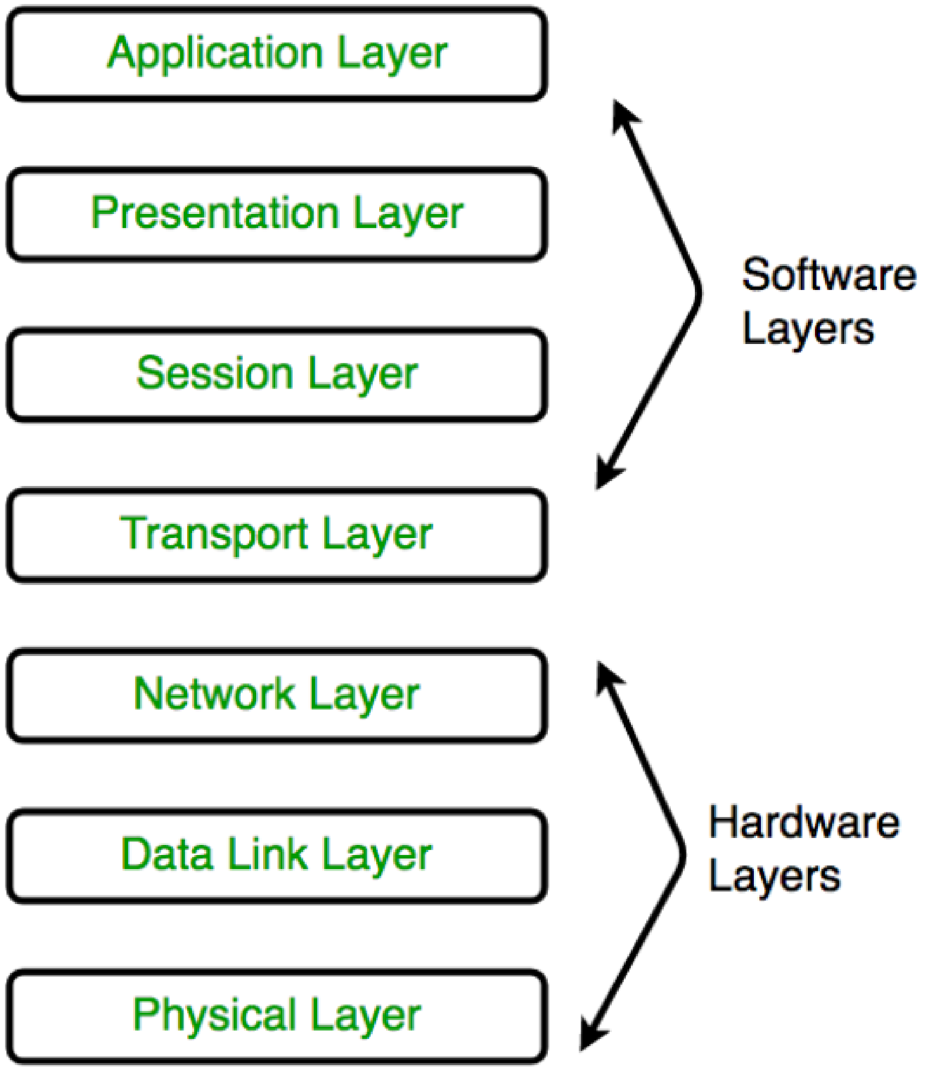


General troubleshooting flow



Using layered models for troubleshooting

- After gathering all the symptoms, the network administrator can compare the characteristics of the problem to the logical layers of the network in order to isolate and resolve the problem.
- Logical networking models, such as the OSI and TCP/IP models, separate network functionality into modular layers. As a rule, hardware problems associated with two or three lower layers OSI model, whereas higher layers describe software ones.
- For example, if the symptoms suggest a physical connection problem, the network administrator can focus on troubleshooting the circuit that operates at the physical layer.



Linux monitoring and troubleshooting tools

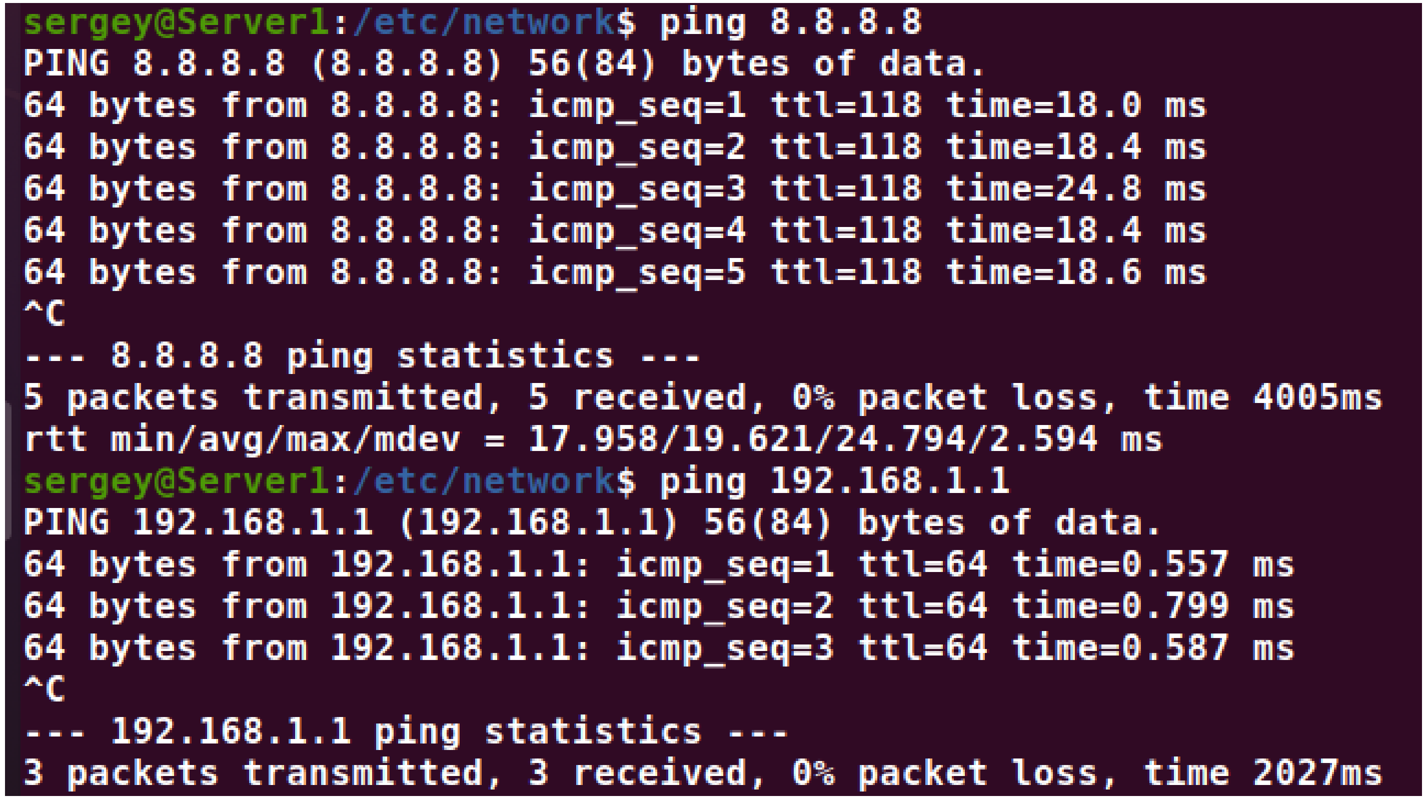
Network monitoring and troubleshooting tools

- ping
- traceroute
- mtr
- netstat
- dig
- nmap
- tcpdump

Ping utility

- The Ping utility is an online free tool that help you to verify if a domain/server is operating and network accessible.
- This tool uses the **Internet Control Message Protocol** (ICMP) Echo function as detailed in RFC 792.
- A small packet will be sent through the network to a given IP address (IPv4) or host name. By default, this packet contains **64 bytes**.
- The device that sent the packet then waits and listens for a return packet. If the connections are good and the target domain/server is up, a good return packet will be received.
- Ping can also tell the user the **number of hops** between two targets and the **amount of time** it takes for a packet to make the complete trip.

Ping utility



```
sergey@Server1:/etc/network$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=118 time=18.0 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=118 time=18.4 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=118 time=24.8 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=118 time=18.4 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=118 time=18.6 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 17.958/19.621/24.794/2.594 ms
sergey@Server1:/etc/network$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.557 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.799 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.587 ms
^C
--- 192.168.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2027ms
```

1. **from**: The destination and its IP address. Note that the IP address may be different for a website depending on your geographical location.

2. **icmp_seq=1**: The sequence number of each ICMP packet. Increases by one for every subsequent echo request.

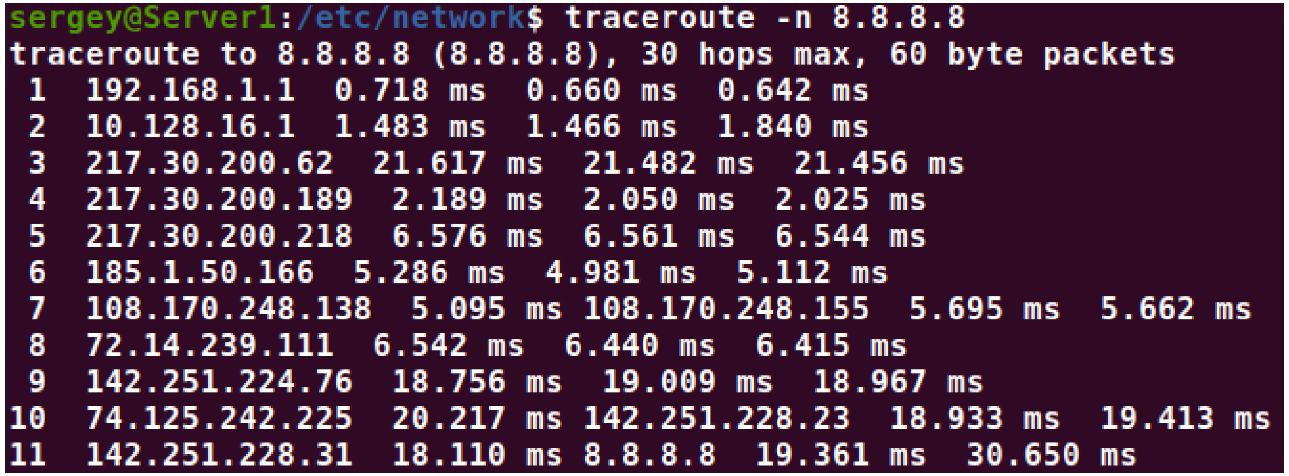
3. **ttl=118**: The Time to Live value from 1 to 255. It represents the number of network hops a packet can take before a router discards it.

4. **time=7.68 ms**: Round trip time (RTT) - the time it took a packet to reach the destination and come back to the source. Expressed in milliseconds.

Traceroute utility

Traceroute is a system administrators' utility to trace the route IP packets take from a source system to some destination system. Traceroute uses the IP TTL parameter to find the route:

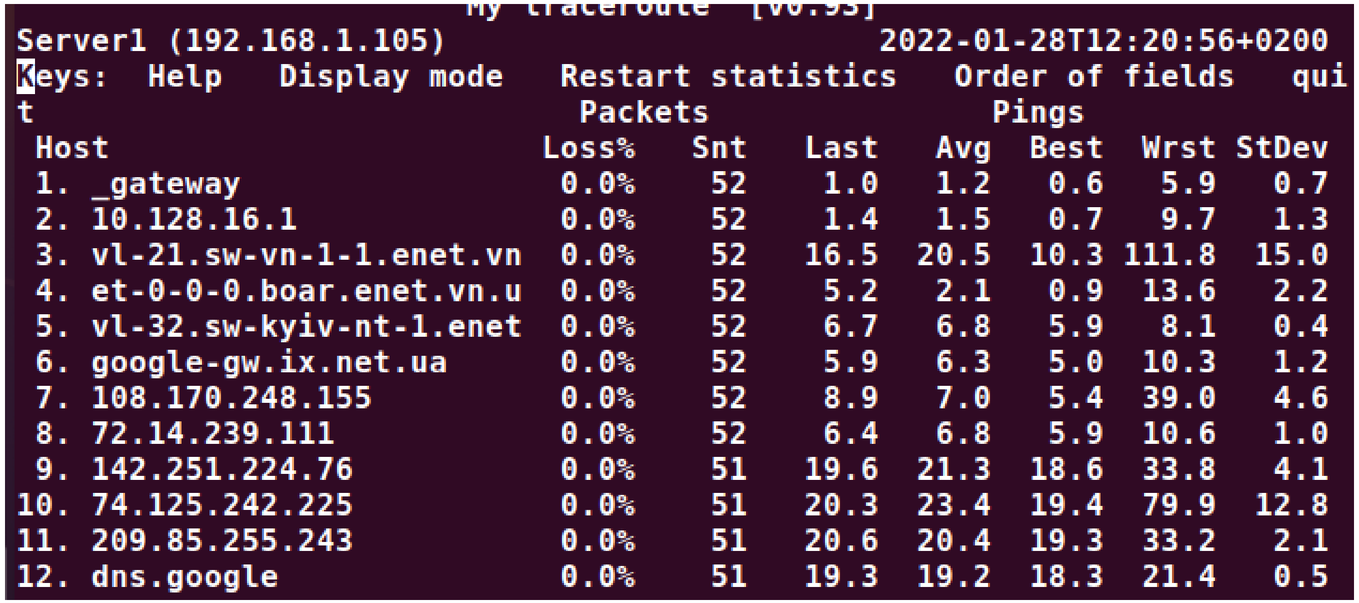
1. It sends a packet with a TTL value equal to 1.
2. The first router receives the packet and decreases the TTL.
3. With a TTL equal to 0, the router sends a timeout back to traceroute, with this packet, traceroute knows about the first router.
4. Now, traceroute sends another packet with a TTL equal to 2.
5. The first router decreases the TTL and sends the packet to the second router which decreases it in turn: the TTL is equal to 0…



```
sergey@Server1:/etc/network$ traceroute -n 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  192.168.1.1  0.718 ms  0.660 ms  0.642 ms
 2  10.128.16.1  1.483 ms  1.466 ms  1.840 ms
 3  217.30.200.62  21.617 ms  21.482 ms  21.456 ms
 4  217.30.200.189  2.189 ms  2.050 ms  2.025 ms
 5  217.30.200.218  6.576 ms  6.561 ms  6.544 ms
 6  185.1.50.166  5.286 ms  4.981 ms  5.112 ms
 7  108.170.248.138  5.095 ms 108.170.248.155  5.695 ms  5.662 ms
 8  72.14.239.111  6.542 ms  6.440 ms  6.415 ms
 9  142.251.224.76  18.756 ms  19.009 ms  18.967 ms
10  74.125.242.225  20.217 ms 142.251.228.23  18.933 ms  19.413 ms
11  142.251.228.31  18.110 ms 8.8.8.8  19.361 ms  30.650 ms
```

mtr utility

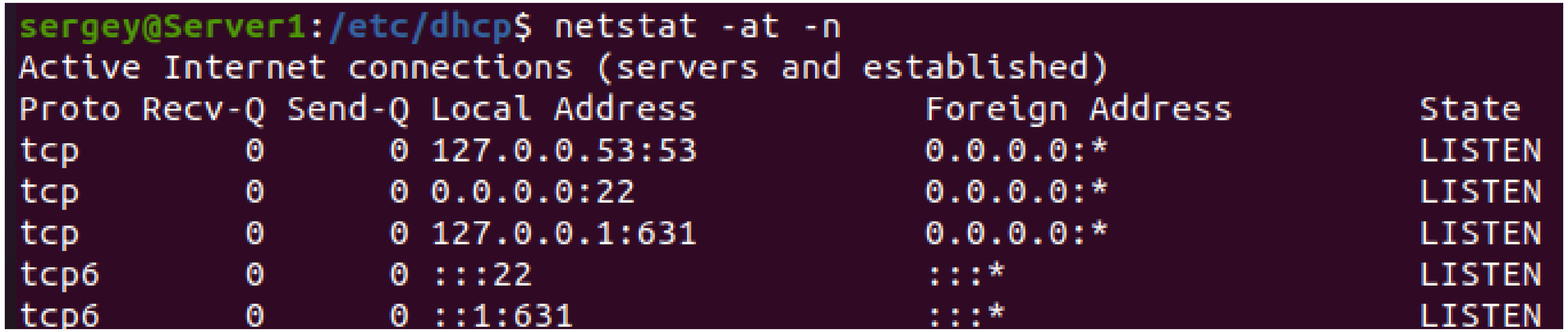
- Mtr (my traceroute) is a command line network diagnostic tool that provides the functionality of both the **ping** and **traceroute** commands.
- Just like a traceroute, the mtr command will show the route from a computer to a specified host. mtr provides a lot of statistics about each hop, such as response time and percentage.
- The mtr command output will display the traceroute report in **real time**.



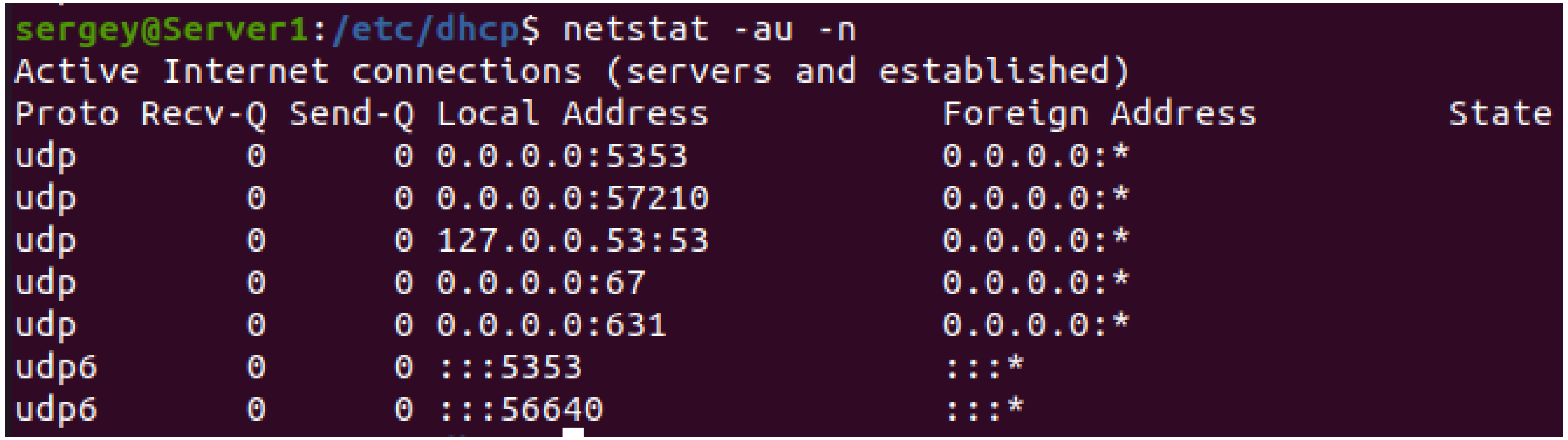
```
                        My traceroute  [v0.93]
Server1 (192.168.1.105)                      2022-01-28T12:20:56+0200
Keys:  Help   Display mode   Restart statistics   Order of fields  qui
t                             Packets               Pings
 Host                       Loss%  Snt   Last   Avg  Best  Wrst StDev
 1. _gateway                 0.0%   52    1.0   1.2   0.6   5.9   0.7
 2. 10.128.16.1              0.0%   52    1.4   1.5   0.7   9.7   1.3
 3. vl-21.sw-vn-1-1.enet.vn  0.0%   52   16.5  20.5  10.3 111.8  15.0
 4. et-0-0-0.boar.enet.vn.u  0.0%   52    5.2   2.1   0.9  13.6   2.2
 5. vl-32.sw-kyiv-nt-1.enet  0.0%   52    6.7   6.8   5.9   8.1   0.4
 6. google-gw.ix.net.ua      0.0%   52    5.9   6.3   5.0  10.3   1.2
 7. 108.170.248.155          0.0%   52    8.9   7.0   5.4  39.0   4.6
 8. 72.14.239.111            0.0%   52    6.4   6.8   5.9  10.6   1.0
 9. 142.251.224.76           0.0%   51   19.6  21.3  18.6  33.8   4.1
10. 74.125.242.225           0.0%   51   20.3  23.4  19.4  79.9  12.8
11. 209.85.255.243           0.0%   51   20.6  20.4  19.3  33.2   2.1
12. dns.google               0.0%   51   19.3  19.2  18.3  21.4   0.5
```

netstat utility

- The Linux netstat command gives you an information about your network connections, the ports that are in use, and the processes using them.



```
sergey@Server1:/etc/dhcp$ netstat -at -n
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 ::1:631                 :::*                    LISTEN
```



```
sergey@Server1:/etc/dhcp$ netstat -au -n
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 0.0.0.0:5353            0.0.0.0:*
udp        0      0 0.0.0.0:57210           0.0.0.0:*
udp        0      0 127.0.0.53:53           0.0.0.0:*
udp        0      0 0.0.0.0:67              0.0.0.0:*
udp        0      0 0.0.0.0:631             0.0.0.0:*
udp6       0      0 :::5353                 :::*
udp6       0      0 :::56640                :::*
```

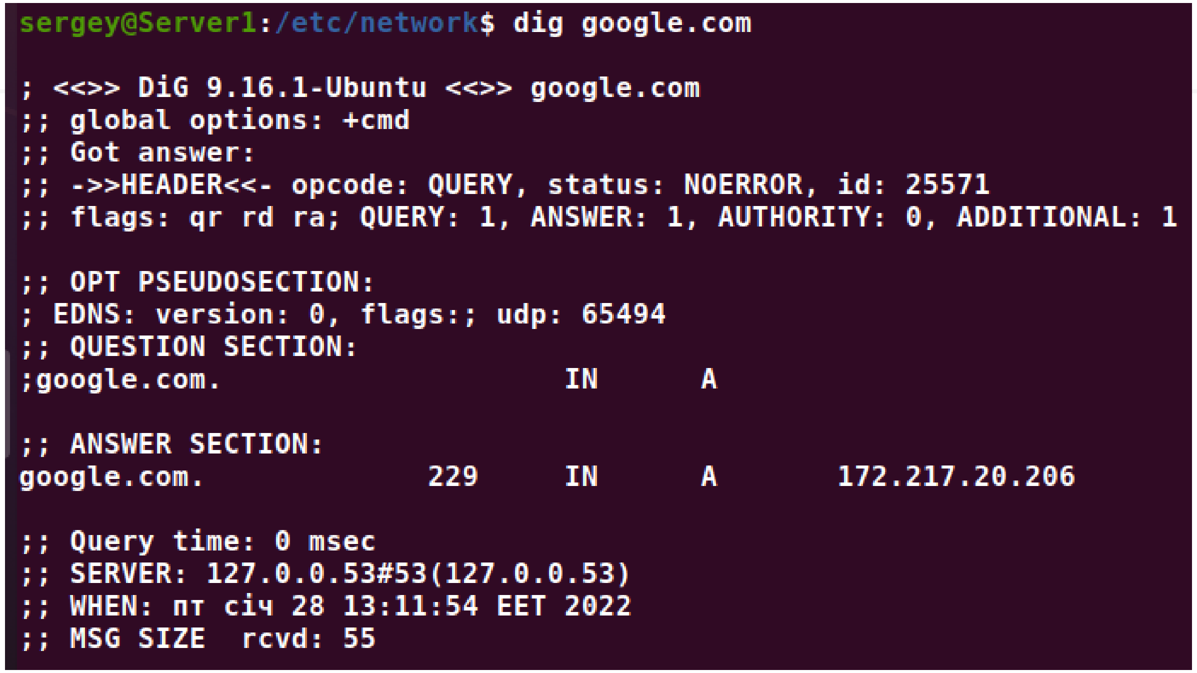
dig and **host** utility

- **Dig** and **host** are used for retrieving information about DNS name servers.
- They are basically used for verifying and troubleshooting DNS problems and to perform DNS lookups.
- Installation: In case of Debian/Ubuntu

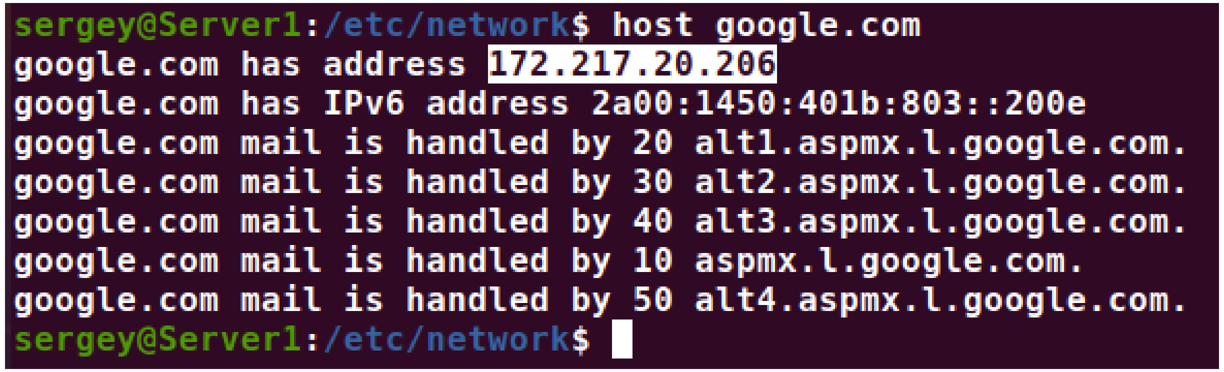
$sudo apt-get install dnsutils

In case of CentOS/RedHat

$sudo yum install bind-utils



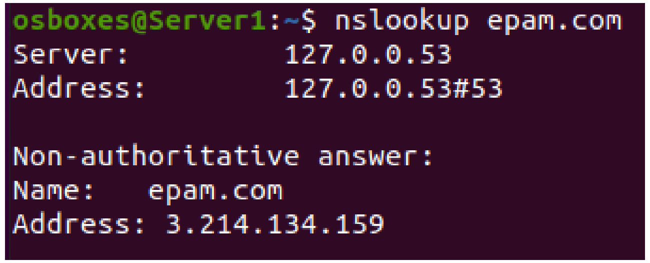
```
sergey@Server1:/etc/network$ dig google.com

; <<>> DiG 9.16.1-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25571
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             229     IN      A       172.217.20.206

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: пт січ 28 13:11:54 EET 2022
;; MSG SIZE  rcvd: 55
```



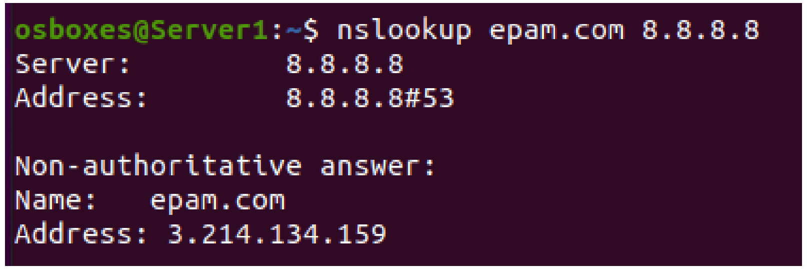
```
sergey@Server1:/etc/network$ host google.com
google.com has address 172.217.20.206
google.com has IPv6 address 2a00:1450:401b:803::200e
google.com mail is handled by 20 alt1.aspmx.l.google.com.
google.com mail is handled by 30 alt2.aspmx.l.google.com.
google.com mail is handled by 40 alt3.aspmx.l.google.com.
google.com mail is handled by 10 aspmx.l.google.com.
google.com mail is handled by 50 alt4.aspmx.l.google.com.
sergey@Server1:/etc/network$
```

Name Resolution *nslookup*

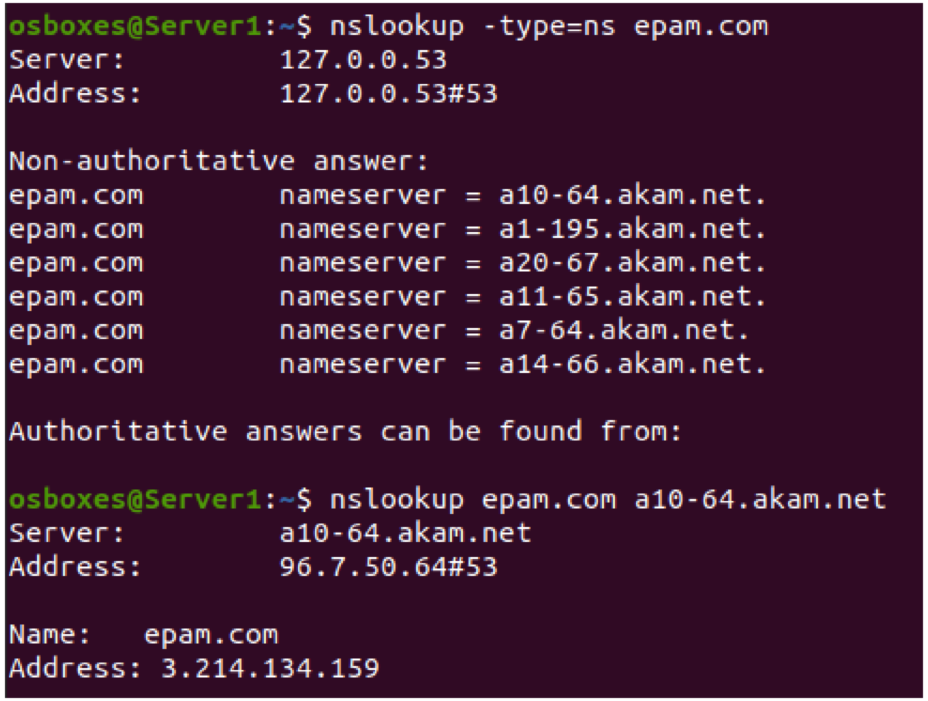
- **nslookup epam.com** – request for default dns server for resolving domain name (epam.com)
- **nslookup epam.com 8.8.8.8** – request for explicit pointed dns server (8.8.8.8) for resolving domain name (epam.com)
- **nslookup -type=ns cisco.com** – request ip address of the domain (epam.com) name servers
- **nslookup 3.214.134.159** - reverse lookup



```
osboxes@Server1:~$ nslookup epam.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   epam.com
Address: 3.214.134.159
```



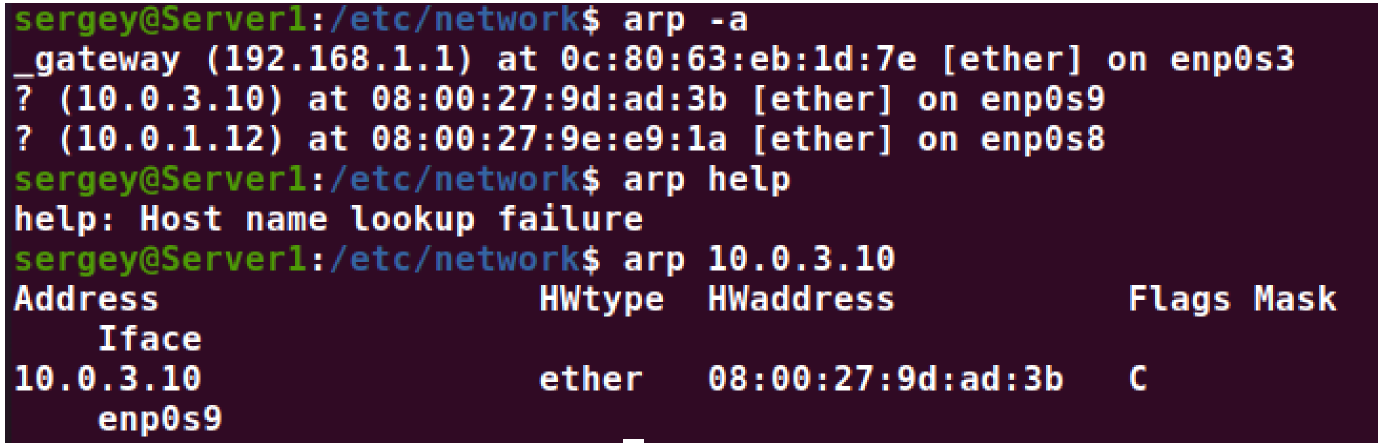
```
osboxes@Server1:~$ nslookup epam.com 8.8.8.8
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   epam.com
Address: 3.214.134.159
```



```
osboxes@Server1:~$ nslookup -type=ns epam.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
epam.com        nameserver = a10-64.akam.net.
epam.com        nameserver = a1-195.akam.net.
epam.com        nameserver = a20-67.akam.net.
epam.com        nameserver = a11-65.akam.net.
epam.com        nameserver = a7-64.akam.net.
epam.com        nameserver = a14-66.akam.net.

Authoritative answers can be found from:

osboxes@Server1:~$ nslookup epam.com a10-64.akam.net
Server:         a10-64.akam.net
Address:        96.7.50.64#53

Name:   epam.com
Address: 3.214.134.159
```

arp utility

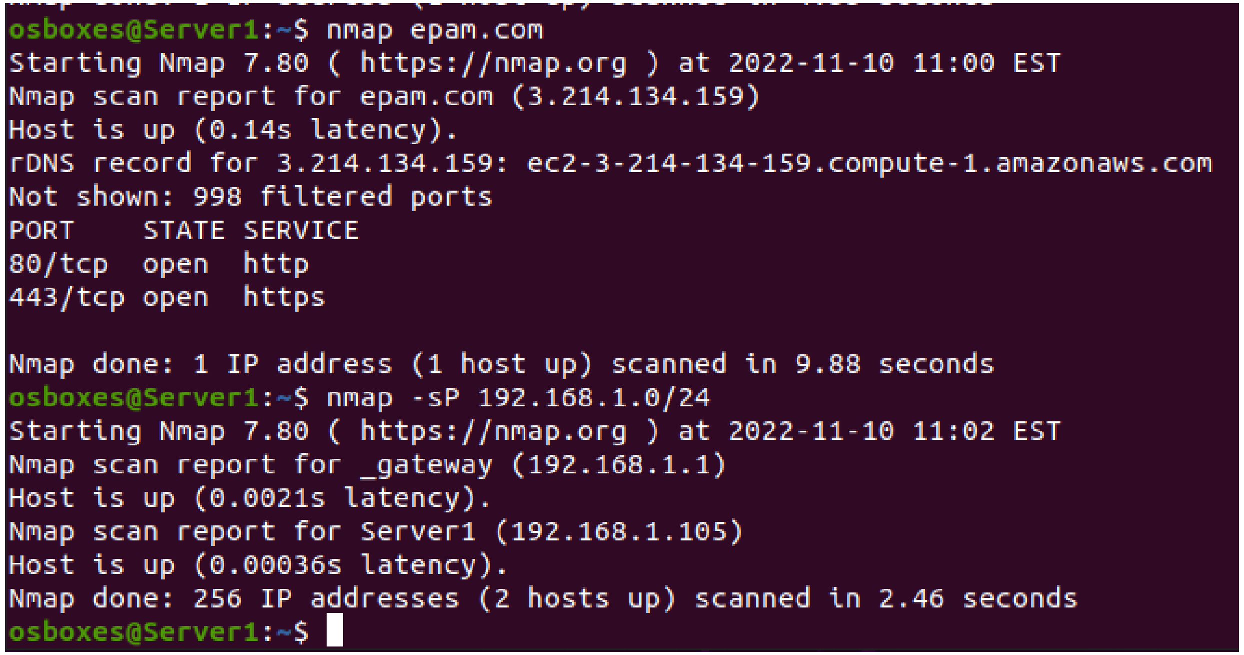
- To display the ARP table on a Unix system, just type "*arp -a*" (this same command will show the arp table in the command prompt on a Windows box).
- The output from arp -a will list the network interface, target system and physical (MAC) address of each system.



```
sergey@Server1:/etc/network$ arp -a
_gateway (192.168.1.1) at 0c:80:63:eb:1d:7e [ether] on enp0s3
? (10.0.3.10) at 08:00:27:9d:ad:3b [ether] on enp0s9
? (10.0.1.12) at 08:00:27:9e:e9:1a [ether] on enp0s8
sergey@Server1:/etc/network$ arp help
help: Host name lookup failure
sergey@Server1:/etc/network$ arp 10.0.3.10
Address                  HWtype  HWaddress           Flags Mask
    Iface
10.0.3.10                ether   08:00:27:9d:ad:3b   C
    enp0s9
```

Nmap utility

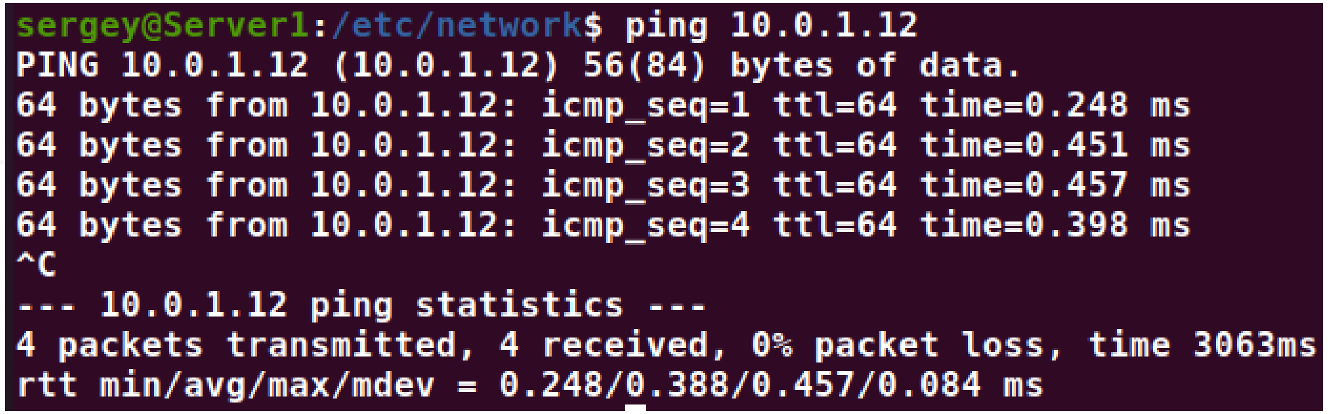
- Nmap, or Network Mapper, is an open source Linux command line tool for network exploration and security auditing.
- With Nmap, server administrators can quickly reveal hosts and services, search for security issues, and scan for open ports.
- The Nmap tool can audit and discover local and remote open ports, as well as network information and hosts.
- To install: *$sudo apt install nmap*
- Help – *nmap -h*
- Attention - Some countries and companies consider port scanning illegal!



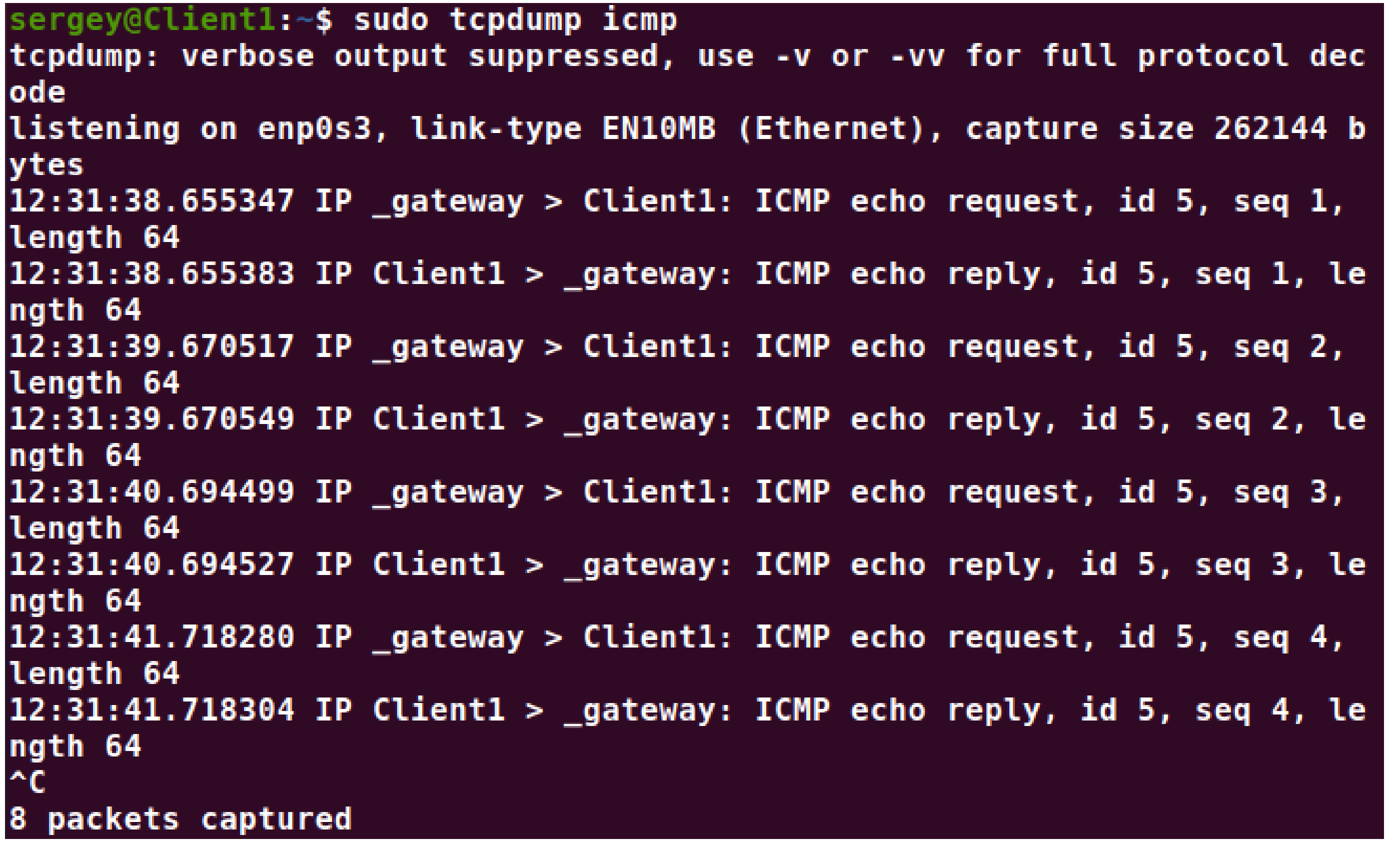
```
osboxes@Server1:~$ nmap epam.com
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-10 11:00 EST
Nmap scan report for epam.com (3.214.134.159)
Host is up (0.14s latency).
rDNS record for 3.214.134.159: ec2-3-214-134-159.compute-1.amazonaws.com
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 9.88 seconds
osboxes@Server1:~$ nmap -sP 192.168.1.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-10 11:02 EST
Nmap scan report for _gateway (192.168.1.1)
Host is up (0.0021s latency).
Nmap scan report for Server1 (192.168.1.105)
Host is up (0.00036s latency).
Nmap done: 256 IP addresses (2 hosts up) scanned in 2.46 seconds
osboxes@Server1:~$ 
```

tcpdump utility

- tcpdump is a most powerful and widely used command-line packets **sniffer** or package analyzer tool which is used to capture or filter TCP/IP packets that are received or transferred over a network on a specific interface.
- It is available under most of the Linux/Unix-based operating systems.
- tcpdump also gives us an option to save captured packets in a file for future analysis in a **pcap** format, that can be viewed by tcpdump command or an open-source GUI-based tool called Wireshark



```
sergey@Server1:/etc/network$ ping 10.0.1.12
PING 10.0.1.12 (10.0.1.12) 56(84) bytes of data.
64 bytes from 10.0.1.12: icmp_seq=1 ttl=64 time=0.248 ms
64 bytes from 10.0.1.12: icmp_seq=2 ttl=64 time=0.451 ms
64 bytes from 10.0.1.12: icmp_seq=3 ttl=64 time=0.457 ms
64 bytes from 10.0.1.12: icmp_seq=4 ttl=64 time=0.398 ms
^C
--- 10.0.1.12 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3063ms
rtt min/avg/max/mdev = 0.248/0.388/0.457/0.084 ms
```



```
sergey@Client1:~$ sudo tcpdump icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol dec
ode
listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 b
ytes
12:31:38.655347 IP _gateway > Client1: ICMP echo request, id 5, seq 1,
length 64
12:31:38.655383 IP Client1 > _gateway: ICMP echo reply, id 5, seq 1, le
ngth 64
12:31:39.670517 IP _gateway > Client1: ICMP echo request, id 5, seq 2,
length 64
12:31:39.670549 IP Client1 > _gateway: ICMP echo reply, id 5, seq 2, le
ngth 64
12:31:40.694499 IP _gateway > Client1: ICMP echo request, id 5, seq 3,
length 64
12:31:40.694527 IP Client1 > _gateway: ICMP echo reply, id 5, seq 3, le
ngth 64
12:31:41.718280 IP _gateway > Client1: ICMP echo request, id 5, seq 4,
length 64
12:31:41.718304 IP Client1 > _gateway: ICMP echo reply, id 5, seq 4, le
ngth 64
^C
8 packets captured
```

Thank you!