MHPC
Master in High Performance Computing

Moreno Baricevic

CNR-IOM DEMOCRITOS
Trieste, ITALY

INTRO TO NETWORKING

PART 2: LINUX commands
(full)

SISSA
ma per seguir virtute e canoscenza
Scuola Internazionale Superiore di Studi Avanzati

ICTP
The Abdus Salam International Centre for Theoretical Physics

Agenda

- Network Interfaces
- LINUX command line utilities
  - Hardware Diagnostic
  - Configuration
  - Software Diagnostic
  - Clients Applications
  - Benchmarking
- Examples

Network Interfaces

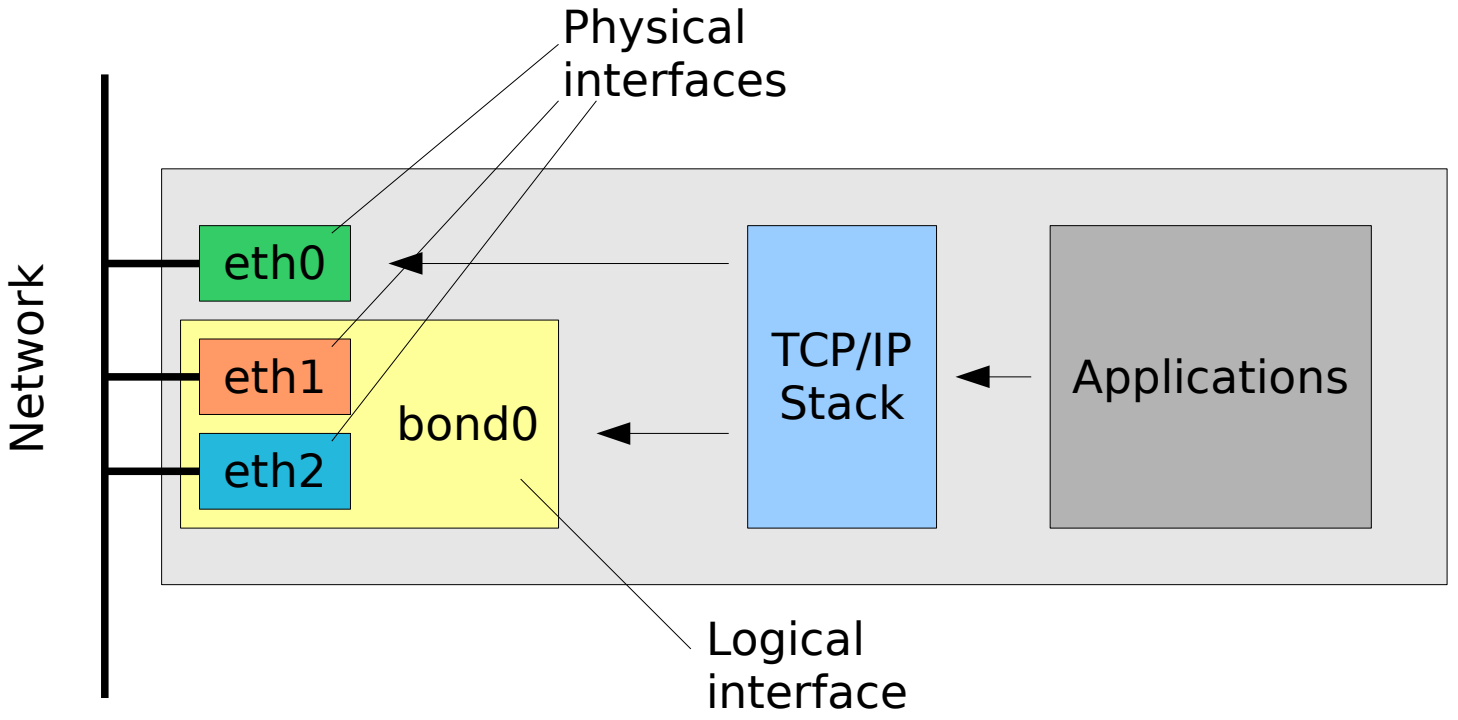
Main network interfaces:

- **lo**: loopback virtual interface for internal networking (provides networking stack to applications). NEVER play with this interface.
- **ethX (eth0, eth1, ...)**: physical Ethernet interfaces
- **ethX:LABEL (eth0:foo, eth0:10, ...)**: virtual interface, in case two or more IP addresses/networks are needed on the same physical interface
- **wlanX** or **iwX (wlan0, ...)**: wireless interface

Interfaces for specific uses:

- **bondX (bond0)**: bonding interface (link aggregation, load balancing), enslave 2 or more interfaces
- **brX (br0)**: ethernet bridging interface (layer 2 forwarding), enslave 2 or more interfaces
- **tunX/tapX (tun0/tap0)**: user-space logical interfaces (virtual machines, tunnels, ...)
- **sit0**: virtual interface which tunnels IPv6-in-IPv4
- (pppX, slipX, bnepX and many many more...)

LINUX Network Stack (example)



eth0: has it's own MAC and IP address, configured as usual

bond0:
- forces the **same** MAC address on both the slaves (**eth1** and **eth2**);
- the MAC address used is the one of the first interface enslaved;
- the IP address belongs to bond0, not eth* (ifconfig bond0 ...);
- depending on the bonding mode adopted, additional configuration may be required on the switch.

Some command line utilities

mii-tool, **ethtool**:	HW diagnostic/configuration
ifconfig, **ip**, **route**:	SW configuration
netstat, **lsof**:	report network resources status
{arp,}ping, **{tcp,}traceroute**:	diagnostic tools
telnet:	simple TCP client
nmap, **nc** (netcat):	TCP/IP swiss army knives
ssh, **scp**, **sftp**:	SSH clients
wget, **curl**:	web downloader (http, ftp, tftp)
tftp, **ftp**:	TFTP and FTP clients
dhclient, **dhcpcd**, **udhcpc**, **pump**:	DHCP clients
nslookup, **host**, **dig**:	DNS clients
tcpdump, **{wire,t}shark**:	network sniffers
iptables, **iptables-save**:	firewall configuration

Hardware Diagnostic

- **mii-tool**: this utility checks or sets the status of a network interface's Media Independent Interface (MII) unit. The default short output reports the negotiated link speed and link status for each interface.

  # mii-tool eth0

  # mii-tool -w

- **ethtool**: display or change ethernet card settings. Is used for querying settings of an ethernet device and changing them. With a single argument specifying the device name prints current setting of the specified device.

  # ethtool eth0
  # ethtool -i eth0

Configuration

- **ifconfig**: is used to configure the kernel-resident network interfaces. It is used at boot time to set up interfaces as necessary. After that, it is usually only needed when debugging or when system tuning is needed.

```
# ifconfig
# ifconfig eth0 192.168.0.2 netmask 255.255.255.0 up
# ifconfig eth0 down
```

- **ip**: show / manipulate routing, devices, policy routing and tunnels

```
# ip addr
# ip link show eth0
# ip monitor link
# ip neigh
```

- **route**: manipulates the kernel's IP routing tables. Its primary use is to set up static routes to specific hosts or networks via an interface after it has been configured with the **ifconfig** program.

```
# route add default gw 192.168.0.1
# route -n
```

Software Diagnostic

- **ping**: uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway
  # ping 127.0.0.1
  # ping 192.168.0.1
  # ping -c 1 -w 10 www.google.com
- **arp**: manipulate the system ARP cache
  # arp -n
- **arping**: send ARP REQUEST to a neighbor host
  # arping 192.168.0.1
  # arping -c 1 -I eth2 192.168.0.1
- **traceroute**: utilizes the IP protocol 'time to live' field and attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to some host
  # traceroute www.google.com
- **tcptraceroute**: **traceroute** implementation using TCP packets
  # tcptraceroute www.google.com

Clients Applications

- **telnet**: user interface to the TELNET protocol, but can be used to open a TCP connection to any port (useful for testing/diagnostic)
  ```
  # telnet switch01
  # telnet www.google.com 80
  ```
- **netcat/nc**: TCP/IP swiss army knife
  ```
  # nc -h
  ```
- **ssh/scp/sftp**: OpenSSH clients (secure shell for remote login, remote file copy and and secure file transfer)
  ```
  # ssh user@ssh.somedomain.com
  # ssh -l user ssh.somedomain.com
  # scp /home/foo/file1 user@hostX.somedomain.com:/tmp/
  ```
- **ftp/tftp**: file transfer programs, FTP and TFTP clients
  ```
  # ftp ftp.somedomain.com
  # tftp -v master.hpc -c get /pxe/pxelinux.0 ./pxelinux0
  ```

Clients Applications

- **wget**: network downloader
  # wget http://www.google.com
  # wget -r -l0 -t0 -np -nc -p -k www.somedomain.com/foo/
- **curl**: transfer data from/to a server using one of the supported protocols (HTTP, HTTPS, FTP, TFTP, DICT, TELNET, LDAP or FILE)
  # curl www.google.com
  # curl tftp://master.hpc/pxe/pxelinux.0 -o /tmp/foo.0
- **links/lynx/w3m**: text-based Web browsers and pages
  # w3m www.google.com

DNS Clients

- **nslookup**: is a program to query Internet domain name servers (uses /etc/resolv.conf for default domain names and servers)

```
# nslookup 192.168.0.1
# nslookup www.google.com
# nslookup www.google.com dns.somedomain.com
```

- **host**: a simple utility for performing DNS lookups. It is normally used to convert names to IP addresses and vice versa.

```
# host 192.168.0.1
# host www.google.com
# host -t MX gmail.com
```

- **dig**: (domain information groper) is a flexible tool for interrogating DNS name servers. DNS administrators use dig to troubleshoot DNS problems because of its flexibility, ease of use and clarity of output. Other lookup tools tend to have less functionality than dig.

```
# dig -x 192.168.0.1
# dig www.google.com
# dig +search www
# dig -t AXFR somedomain.com
```

DHCP clients

- **dhclient**: the Internet Systems Consortium DHCP Client provides a means for configuring one or more network interfaces using the Dynamic Host Configuration Protocol, BOOTP protocol, or if these protocols fail, by statically assigning an address.
  ```
  # dhclient eth0
  # dhclient -n eth0
  ```
- **dhcpcd**: is a DHCP client daemon
  ```
  # dhcpcd eth0
  # dhcpcd -R -N -t 60 eth0
  ```
- **pump**: yet another DHCP client (debian/ubuntu/knoppix specific)
- **udhcpc**: micro DHCP client, provided by **busybox**
  ```
  # udhcpc -i eth0 -f -n -q
  ```

Network Resources Status

- **netstat**: print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships
  ```
  # netstat -p -u -t -a -n
  # netstat -rn
  ```
- **lsof**: list open files and sockets (and a lot of other things)
  ```
  # lsof -nP -i TCP -a -c ssh
  # lsof -nP -i UDP
  ```
- **fuser**: identify processes using files or sockets
  ```
  # fuser -v -n tcp 22
  ```
- **ss**: yet another utility to investigate sockets
  ```
  # ss -4 -n -a
  ```

Network Sniffing and Monitoring and...

- **tcpdump**: dump traffic on a network (sniffer)
  ```
  # tcpdump -i eth0 -nn
  # tcpdump -i any -qtep port bootpc and ip broadcast
  # tcpdump -i any -e arp or icmp
  ```
- **tshark/wireshark**: dump and analize network traffic (providing also a graphic interface)
  ```
  # wireshark &
  # tshark -i eth0 -V arp
  ```
- **ettercap**: sniffing of live connections, content filtering, active and passive dissection of many protocols
- **arpwatch**: keep track of ethernet/ip address pairings (logs activity and reports certain changes via e-mail)

Firewall Configuration and Testing

- **iptables**: administration tool for IPv4 packet filtering and NAT

```
# iptables -A INPUT -m state \
  –state ESTABLISHED,RELATED -j ACCEPT
# iptables -A INPUT -p tcp --dport 22 --syn -j ACCEPT
# iptables -A INPUT -j DROP
# iptables -nvL INPUT
```

- **iptables-save/iptables-restore**: show, save and restore iptables configuration

```
# iptables-save | grep '\-A INPUT' | nl
# iptables-save > ./iptables.conf
# iptables-restore < ./iptables.conf
```

- **nmap**: network exploration tool and security / port scanner

```
# nmap -sP 192.168.0.0/24
# nmap -sS -p 22,25,80,443,8080 hostX
```

- **netcat/nc, telnet, ping, arping, hping2, tcptraceroute, ...**: file transfer programs, FTP and TFTP clients

Some network benchmarking tools

- **iperf**
  - http://iperf.sourceforge.net/
- **netperf**
  - http://www.netperf.org/
- **netpipe**
  - http://www.scl.ameslab.gov/Projects/NetPIPE/

Command line examples

- diagnose hardware connection
- network configuration
- diagnose local networking
- diagnose remote networking
- diagnose high level apps
- traffic sniffing

Diagnostic – Connection/HW no link – interface down

```
[root@localhost:~]# mii-tool eth0
eth0: no link

[root@localhost:~]# ethtool -i eth0
driver: 3c59x
version:
firmware-version:
bus-info: 0000:02:01.0

[root@localhost:~]# ethtool eth0
Settings for eth0:
        Supported ports: [ TP MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
        Supports auto-negotiation: Yes
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
        Advertised auto-negotiation: Yes
        Speed: 10Mb/s
        Duplex: Half
        Port: MII
        PHYAD: 2
        Transceiver: internal
        Auto-negotiation: on
        Current message level: 0x00000001 (1)
        Link detected: no

[root@localhost:~]# ip link show eth0
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:26:54:0c:1e:b1 brd ff:ff:ff:ff:ff:ff

[root@localhost:~]# ifconfig eth0
eth0    Link encap:Ethernet  HWaddr 00:26:54:0C:1E:B1
        BROADCAST MULTICAST  MTU:1500  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
        Interrupt:10 Base address:0xc000
```

Diagnostic – Connection/HW no link – interface up

```
[root@localhost:~]# ifconfig 192.168.10.1 netmask 255.255.255.252 up
[root@localhost:~]# mii-tool eth0
eth0: no link
[root@localhost:~]# ethtool eth0
Settings for eth0:
        Supported ports: [ TP MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
        Supports auto-negotiation: Yes
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
        Advertised auto-negotiation: Yes
        Speed: 10Mb/s
        Duplex: Half
        Port: MII
        PHYAD: 2
        Transceiver: internal
        Auto-negotiation: on
        Current message level: 0x00000001 (1)
        Link detected: no
[root@localhost:~]# ip link show eth0
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:26:54:0c:1e:b1 brd ff:ff:ff:ff:ff:ff
[root@localhost:~]# ifconfig eth0
eth0    Link encap:Ethernet  HWaddr 00:26:54:0C:1E:B1
        inet addr:192.168.10.1  Bcast:192.168.10.3  Mask:255.255.255.252
        UP BROADCAST MULTICAST  MTU:1500  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
        Interrupt:10 Base address:0x2000
```

Diagnostic – Connection/HW link ok – interface up

```
[root@localhost:~]# mii-tool eth0
eth0: negotiated 100baseTx-FD, link ok
[root@localhost:~]# ethtool eth0
Settings for eth0:
        Supported ports: [ TP MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
        Supports auto-negotiation: Yes
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
        Advertised auto-negotiation: Yes
        Speed: 100Mb/s
        Duplex: Full
        Port: MII
        PHYAD: 2
        Transceiver: internal
        Auto-negotiation: on
        Current message level: 0x00000001 (1)
        Link detected: yes
[root@localhost:~]# ip link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:26:54:0c:1e:b1 brd ff:ff:ff:ff:ff:ff
[root@localhost:~]# ifconfig eth0
eth0    Link encap:Ethernet  HWaddr 00:26:54:0C:1E:B1
        inet addr:192.168.10.1  Bcast:192.168.10.3  Mask:255.255.255.252
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:3 errors:0 dropped:0 overruns:0 frame:0
        TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:256 (256.0 b)  TX bytes:724 (724.0 b)
        Interrupt:10 Base address:0x6000
```

Diagnostic – Connection/HW monitoring link

```
[root@localhost:~]# ip monitor link
4: eth2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast
    link/ether 00:0e:0c:c1:78:6c brd ff:ff:ff:ff:ff:ff
4: eth2: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast
    link/ether 00:0e:0c:c1:78:6c brd ff:ff:ff:ff:ff:ff
^C
[root@localhost:~]# mii-tool --watch eth2
09:54:25 eth2: negotiated 100baseTx-FD flow-control, link ok
09:54:30 eth2: no link
09:54:32 eth2: negotiated 100baseTx-FD flow-control, link ok
^C
[root@localhost:~]# dmesg | grep 'NIC Link'
[49963.915709] e1000: eth2 NIC Link is Down
[49968.157615] e1000: eth2 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX/TX
[root@localhost:~]# while sleep 1 ; do echo -n `date` ; ethtool eth2 | grep Link ; done
Mon Nov 16 09:54:28 CET 2009    Link detected: yes
Mon Nov 16 09:54:29 CET 2009    Link detected: yes
Mon Nov 16 09:54:30 CET 2009    Link detected: no
Mon Nov 16 09:54:31 CET 2009    Link detected: no
Mon Nov 16 09:54:32 CET 2009    Link detected: yes
Mon Nov 16 09:54:33 CET 2009    Link detected: yes
^C
```

Diagnostic – Configuration (show)

```
[user@localhost:~]$ /sbin/ifconfig
eth0  Link encap:Ethernet  HWaddr 00:26:54:0C:1E:B1
      inet addr:192.168.10.1  Bcast:192.168.10.3  Mask:255.255.255.252
      inet6 addr: fe80::226:54ff:fe0c:1eb1/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:6 errors:0 dropped:0 overruns:0 frame:0
      TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:512 (512.0 b)  TX bytes:980 (980.0 b)
      Interrupt:10 Base address:0x2000

lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING  MTU:16436  Metric:1
      RX packets:6419 errors:0 dropped:0 overruns:0 frame:0
      TX packets:6419 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:692969 (676.7 KiB)  TX bytes:692969 (676.7 KiB)

[user@localhost:~]$ /sbin/ip addr
1: lo: <LOOPBACK,UP,10000> mtu 16436 qdisc noqueue
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
   inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
   link/ether 00:26:54:0c:1e:b1 brd ff:ff:ff:ff:ff:ff
   inet 192.168.10.1/30 brd 192.168.10.3 scope global eth0
   inet6 fe80::226:54ff:fe0c:1eb1/64 scope link
      valid_lft forever preferred_lft forever
```

Diagnostic – Configuration (set) IP binding and routing

```
[user@localhost:~]$ /sbin/ifconfig eth0 192.168.10.1 netmask 255.255.0.0 broadcast 192.168.255.255 mtu 1500 up

[user@localhost:~]$ /sbin/ip address add dev eth0 192.168.10.1/16 br +
```

```
[user@localhost:~]$ /sbin/route add default gw 192.168.0.1

[user@localhost:~]$ /sbin/route add -net 10.0.0.0/8 dev eth0

[user@localhost:~]$ /sbin/route add -host 172.16.0.1 gw 192.168.0.2

[user@localhost:~]$ /sbin/route add -host 172.16.0.1 gw 192.168.0.3 metric 10

[user@localhost:~]$ /sbin/route add -host 239.2.11.71 dev eth0

[user@localhost:~]$ /sbin/route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags   Metric  Ref  Use  Iface
239.2.11.71     0.0.0.0         255.255.255.255 UH      0       0    0    eth0
172.16.0.1      192.168.0.2     255.255.255.255 UGH     0       0    0    eth0
172.16.0.1      192.168.0.3     255.255.255.255 UGH     10      0    0    eth0
192.168.0.0     0.0.0.0         255.255.0.0     U       0       0    0    eth0
10.0.0.0        0.0.0.0         255.0.0.0       U       0       0    0    eth0
127.0.0.0       0.0.0.0         255.0.0.0       U       0       0    0    lo
0.0.0.0         192.168.0.1     0.0.0.0         UG      0       0    0    eth0
```

Diagnostic – Configuration (set) Advanced and useless routing...

```
[user@localhost:~]$ /sbin/ip route add to blackhole 192.168.24.0/24
[user@localhost:~]$ /sbin/ip route add to prohibit 192.168.0.201
[user@localhost:~]$ /sbin/ip route add to unreachable 192.168.10.99
[user@localhost:~]$ /sbin/ip route add to 99.99.99.0/24 dev eth0
[user@localhost:~]$ /sbin/ip route add to 99.99.0.0/24 via 99.99.99.1 metric 10
[user@localhost:~]$ /sbin/ip route add to local 192.0.2.0/24 dev lo

[user@localhost:~]$ /sbin/route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.0.1      192.168.0.2     255.255.255.255 UGH   0      0        0 eth0
172.16.0.1      192.168.0.1     255.255.255.255 UGH   10     0        0 eth0
192.168.0.201   -               255.255.255.255 !H    0      -        0 -
192.168.10.99   -               255.255.255.255 !H    0      -        0 -
239.2.11.71     0.0.0.0         255.255.255.255 UH    0      0        0 eth0
99.99.0.0       99.99.99.1      255.255.255.0   UG    0      0        0 eth0
99.99.99.0      0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.24.0    0.0.0.0         255.255.255.0   U     0      0        0 *
10.1.0.0        0.0.0.0         255.255.0.0     U     0      0        0 eth0
192.168.0.0     0.0.0.0         255.255.0.0     U     0      0        0 eth0
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 eth2
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
0.0.0.0         192.168.0.1     0.0.0.0         UG    0      0        0 eth0
[user@localhost:~]$ /sbin/ip route
172.16.0.1 via 192.168.0.2 dev eth0
172.16.0.1 via 192.168.0.1 dev eth0  metric 10
prohibit 192.168.0.201
unreachable 192.168.10.99
239.2.11.71 dev eth0  scope link
99.99.0.0/24 via 99.99.99.1 dev eth0
99.99.99.0/24 dev eth0  scope link
blackhole 192.168.24.0/24
10.1.0.0/16 dev eth0  scope link
192.168.0.0/16 dev eth0  proto kernel  scope link  src 192.168.10.1
10.0.0.0/8 dev eth2  proto kernel  scope link  src 10.0.0.1
127.0.0.0/8 dev lo  scope link
default via 192.168.0.1 dev eth0
```

Diagnostic – Configuration (set) Advanced and useless routing...

```
[user@localhost:~]$ /sbin/ip route get 192.168.0.2
192.168.0.2 dev eth0  src 192.168.10.1
    cache  mtu 1500 advmss 1460 metric 10 64
[user@localhost:~]$ /sbin/ip route get 192.168.24.1
RTNETLINK answers: Network is unreachable
[user@localhost:~]$ /sbin/ip route get 192.168.0.201
RTNETLINK answers: Network is unreachable
[user@localhost:~]$ /sbin/ip route get 192.168.10.99
RTNETLINK answers: Network is unreachable
[user@localhost:~]$ /sbin/ip route get 99.99.99.1
99.99.99.1 dev eth0  src 192.168.10.1
    cache  mtu 1500 advmss 1460 metric 10 64
[user@localhost:~]$ /sbin/ip route get 99.99.0.1
99.99.0.1 via 99.99.99.1 dev eth0  src 192.168.10.1
    cache  mtu 1500 advmss 1460 metric 10 64
[user@localhost:~]$ /sbin/ip route get 192.0.2.1
local 192.0.2.1 dev lo  src 192.0.2.1
    cache <local>  mtu 16436 advmss 16396 metric 10 64
[user@localhost:~]$ /sbin/ip route get 1.1.1.1
1.1.1.1 via 192.168.0.1 dev eth0  src 192.168.10.1
    cache  mtu 1500 advmss 1460 metric 10 64
```

Diagnostic - Local networking

```
[user@localhost:~]$ ping -c1 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.058 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.058/0.058/0.058/0.000 ms
```

```
[user@localhost:~]$ traceroute 127.0.0.1
traceroute to 127.0.0.1 (127.0.0.1), 30 hops max, 38 byte packets
 1  localhost.localdomain (127.0.0.1)  0.097 ms  0.146 ms  0.030 ms
```

```
[root@localhost:~]# tcptraceroute 127.0.0.1
Selected device lo, address 127.0.0.1, port 43494 for outgoing packets
Tracing the path to 127.0.0.1 on TCP port 80 (http), 30 hops max
 1  localhost.localdomain (127.0.0.1) [closed]  0.079 ms  0.029 ms  0.025 ms
```

Diagnostic - Remote networking (DNS)

```
[user@localhost:~]$ nslookup www.google.com
;; connection timed out; no servers could be reached
[user@localhost:~]$ host www.google.com
;; connection timed out; no servers could be reached
[user@localhost:~]$ dig www.google.com
; <<>> DiG 9.3.2-P1 <<>> www.google.com
;; global options:  printcmd
;; connection timed out; no servers could be reached
```

```
[user@localhost:~]$ nslookup www.google.com
...
[user@localhost:~]$ host www.google.com
...
[user@localhost:~]$ dig www.google.com
...
[user@localhost:~]$ dig +search www
...
```

```
[user@localhost:~]$ nslookup 10.0.0.1
...
[user@localhost:~]$ host 10.0.0.1
...
[user@localhost:~]$ dig -x 10.0.0.1
...
```

Diagnostic - Remote networking (ICMP)

```
[user@localhost:~]$ ping -c1 www.google.com
PING www.l.google.com (209.85.129.99) 56(84) bytes of data.
64 bytes from fk-in-f99.1e100.net (209.85.129.99): icmp_seq=1 ttl=55 time=17.7 ms

--- www.l.google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 17.726/17.726/17.726/0.000 ms
```

```
[user@localhost:~]$ traceroute www.google.com
traceroute: Warning: www.google.com has multiple addresses; using 209.85.129.147
traceroute to www.l.google.com (209.85.129.147), 30 hops max, 40 byte packets
 1  rt-sissa-217 (147.122.255.217)  0.506 ms  0.282 ms  0.295 ms
 2  ru-miramare-rc-ts1.ts1.garr.net (193.206.132.21)  0.630 ms  0.623 ms  0.625 ms
 3  rc-ts1-rt-mi2.mi2.garr.net (193.206.134.205)  8.694 ms  8.615 ms  8.588 ms
 4  193.206.129.130 (193.206.129.130)  8.682 ms  8.620 ms  8.875 ms
 5  216.239.47.128 (216.239.47.128)  8.722 ms 209.85.249.54 (209.85.249.54)  8.796 ms
      216.239.47.128 (216.239.47.128)  8.911 ms
 6  209.85.249.234 (209.85.249.234)  18.223 ms  18.206 ms  18.145 ms
 7  72.14.232.201 (72.14.232.201)  17.940 ms 72.14.232.203 (72.14.232.203)  18.068 ms
      18.037 ms
 8  72.14.233.206 (72.14.233.206)  21.409 ms  21.715 ms 72.14.239.170 (72.14.239.170)
      18.319 ms
 9  fk-in-f147.1e100.net (209.85.129.147)  17.994 ms  18.155 ms  17.967 ms
```

Diagnostic - Remote networking (TCP)

```
[root@localhost:~]# tcptraceroute www.google.com
Selected device eth0, address 147.122.10.31, port 60078 for outgoing packets
Tracing the path to www.google.com (209.85.129.147) on TCP port 80 (www), 30
hops max
 1  rt-sissa-217.sissa.it (147.122.255.217)  0.509 ms  0.295 ms  0.219 ms
 2  ru-miramare-rc-ts1.ts1.garr.net (193.206.132.21)  0.596 ms  0.608 ms  0.578 ms
 3  rc-ts1-rt-mi2.mi2.garr.net (193.206.134.205)  8.645 ms  8.553 ms  11.025 ms
 4  193.206.129.134  8.642 ms  8.646 ms  8.555 ms
 5  209.85.249.54  8.689 ms  8.736 ms  8.760 ms
 6  209.85.251.113  17.333 ms  17.296 ms  17.456 ms
 7  72.14.232.165  17.429 ms  17.471 ms  17.498 ms
 8  72.14.239.170  20.727 ms  17.693 ms  17.968 ms
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  fk-in-f147.1e100.net (209.85.129.147) [open]  17.878 ms  18.084 ms *
```

Diagnostic – Using telnet

```
[root@localhost:~]# telnet www.google.com 80
Trying 209.85.129.103...
Connected to www.google.com (209.85.129.103).
Escape character is '^]'.
GET /
HTTP/1.0 302 Found
Location: http://www.google.it/
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Set-Cookie:
PREF=ID=efe22e8583b659c9:TM=1257415592:LM=1257415592:S=EzQ9uSnMVEoIFao8;
expires=Sat, 05-Nov-2011 10:06:32 GMT; path=/; domain=.google.com
Set-Cookie:
NID=28=qUvEfYMatP4god6U-NaXmgb5sF9VjjtqhHpDyvGA6Hh8qFe6SIvV2cKjp01wCFRGSMQHUs6
MZppPjHMnT7R_7rnADH7eXx75FAe6rERtGM8iUvp3BIImnpDXplVCVqv6; expires=Fri, 07-May-2010
10:06:32 GMT; path=/; domain=.google.com; HttpOnly
Date: Thu, 05 Nov 2009 10:06:32 GMT
Server: gws
Content-Length: 218
X-XSS-Protection: 0

<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="http://www.google.it/">here</A>.
</BODY></HTML>
Connection closed by foreign host.
```

Diagnostic – Using telnet (and netcat)

```
[root@localhost:~]# telnet www.democritos.it 80
Trying 147.122.10.31...
Connected to www.democritos.it (147.122.10.31).
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Thu, 05 Nov 2009 10:13:46 GMT
Server: Apache
Last-Modified: Thu, 02 Jul 2009 14:39:20 GMT
ETag: "af03e-5841-4a4cc698"
Accept-Ranges: bytes
Content-Length: 22593
Connection: close
Content-Type: text/html; charset=iso-8859-1

Connection closed by foreign host.
```

```
[root@localhost:~]# printf 'HEAD / HTTP/1.0\n\n' | nc www.democritos.it 80
HTTP/1.1 200 OK
Date: Thu, 05 Nov 2009 10:15:52 GMT
Server: Apache
Last-Modified: Thu, 02 Jul 2009 14:39:20 GMT
ETag: "af03e-5841-4a4cc698"
Accept-Ranges: bytes
Content-Length: 22593
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

Diagnostic – Using wget

```
[root@localhost:~]# wget -–spider -S www.democritos.it
Spider mode enabled. Check if remote file exists.
--2009-11-05 11:12:08--  http://www.democritos.it/
Resolving www.democritos.it... 147.122.10.31
Connecting to www.democritos.it|147.122.10.31|:80... connected.
HTTP request sent, awaiting response...
  HTTP/1.1 200 OK
  Date: Thu, 05 Nov 2009 10:12:10 GMT
  Server: Apache
  Last-Modified: Thu, 02 Jul 2009 14:39:20 GMT
  ETag: "af03e-5841-4a4cc698"
  Accept-Ranges: bytes
  Content-Length: 22593
  Keep-Alive: timeout=15, max=100
  Connection: Keep-Alive
  Content-Type: text/html; charset=iso-8859-1
Length: 22593 (22K) [text/html]
Remote file exists and could contain further links,
but recursion is disabled -- not retrieving.
```

Diagnostic – Using ARP

```
[root@localhost:~]# cat /proc/net/arp
IP address  HW type    Flags HW address           Mask Device
10.2.1.16         0x1         0x2         00:09:3D:12:1C:C8    *            eth2
10.2.0.58         0x1         0x2         00:30:48:2C:61:E1    *            eth2
10.2.1.17         0x1         0x0         00:09:3D:12:06:92    *            eth2
147.122.17.1      0x1         0x2         00:0B:FD:42:BA:7F    *            eth0
[root@localhost:~]# arp -an
? (10.2.1.16) at 00:09:3D:12:1C:C8 [ether] on eth2
? (10.2.0.58) at 00:30:48:2C:61:E1 [ether] on eth2
? (10.2.1.17) at <incomplete> on eth2
? (147.122.17.1) at 00:0B:FD:42:BA:7F [ether] on eth0
[root@localhost:~]# ip neigh
10.2.1.16 dev eth2 lladdr 00:09:3d:12:1c:c8 nud stale
10.2.0.58 dev eth2 lladdr 00:30:48:2c:61:e1 nud stale
10.2.1.17 dev eth2  nud failed
147.122.17.1 dev eth0 lladdr 00:0b:fd:42:ba:7f nud reachable
```

Diagnostic – Using TCPDUMP/TSHARK

src IP = 192.168.1.1 src MAC = 00:0e:0c:21:fb:f6
dst IP = 192.168.0.101 dst MAC = 00:04:76:9b:ec:46

```
t0  [root@localhost:~]# tshark -i eth0 arp or icmp
t2{ 0.000000 00:0e:0c:21:fb:f6 -> ff:ff:ff:ff:ff:ff ARP Who has 192.168.0.101?  Tell 192.168.1.1
    0.000142 00:04:76:9b:ec:46 -> 00:0e:0c:21:fb:f6 ARP 192.168.0.101 is at 00:04:76:9b:ec:46
t3{ 0.000169 192.168.1.1 -> 192.168.0.101 ICMP Echo (ping) request
    0.000264 192.168.0.101 -> 192.168.1.1 ICMP Echo (ping) reply
```

```
t1  [root@localhost:~]# ping -c 1 192.168.0.101
    PING node101 (192.168.0.101) from 192.168.1.1 : 56(84) bytes of data.
t3  64 bytes from node101 (192.168.0.101): icmp_seq=1 ttl=64 time=0.493 ms

    --- node101 ping statistics ---
    1 packets transmitted, 1 received, 0% loss, time 0ms
    rtt min/avg/max/mdev = 0.493/0.493/0.493/0.000 ms
```

Diagnostic – Network Resources Status

```
[root@localhost:~]# netstat -p -u -t -a -n | head -n 6
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address  Foreign Address  State    PID/Program name
tcp        0      0 0.0.0.0:2049   0.0.0.0:*        LISTEN   -
tcp        0      0 0.0.0.0:961    0.0.0.0:*        LISTEN   3747/rpc.statd
tcp        0      0 0.0.0.0:963    0.0.0.0:*        LISTEN   3750/rpc.rquotad
tcp        0      0 0.0.0.0:37     0.0.0.0:*        LISTEN   3698/inetd
```

```
[root@localhost:~]# lsof -nP -i TCP -a -c ssh
COMMAND  PID USER   FD   TYPE DEVICE SIZE NODE NAME
sshd    3704 root    3u  IPv6   7768       TCP *:22 (LISTEN)
ssh     4181 root    3u  IPv4   8932       TCP 10.0.0.1:49771->10.0.0.2:22 (ESTABLISHED)
ssh     4352 root    3u  IPv4   9113       TCP 10.0.0.1:58678->10.0.0.2:22 (ESTABLISHED)
```

```
[root@localhost:~]# fuser -v -n 22 25 80 443
                     USER        PID ACCESS COMMAND
22/tcp:              root       3704 F.... sshd
```

```
[root@localhost:~]# ss -4 -a | head -n 5
COMMAND  PID USER   FD   TYPE DEVICE SIZE NODE NAME
State      Recv-Q Send-Q      Local Address:Port          Peer Address:Port
LISTEN     0      64                  *:nfsd                     *:*
LISTEN     0      128                 *:961                      *:*
LISTEN     0      128                 *:963                      *:*
LISTEN     0      128                 *:time                     *:*
```

Netfilter (Firewall)

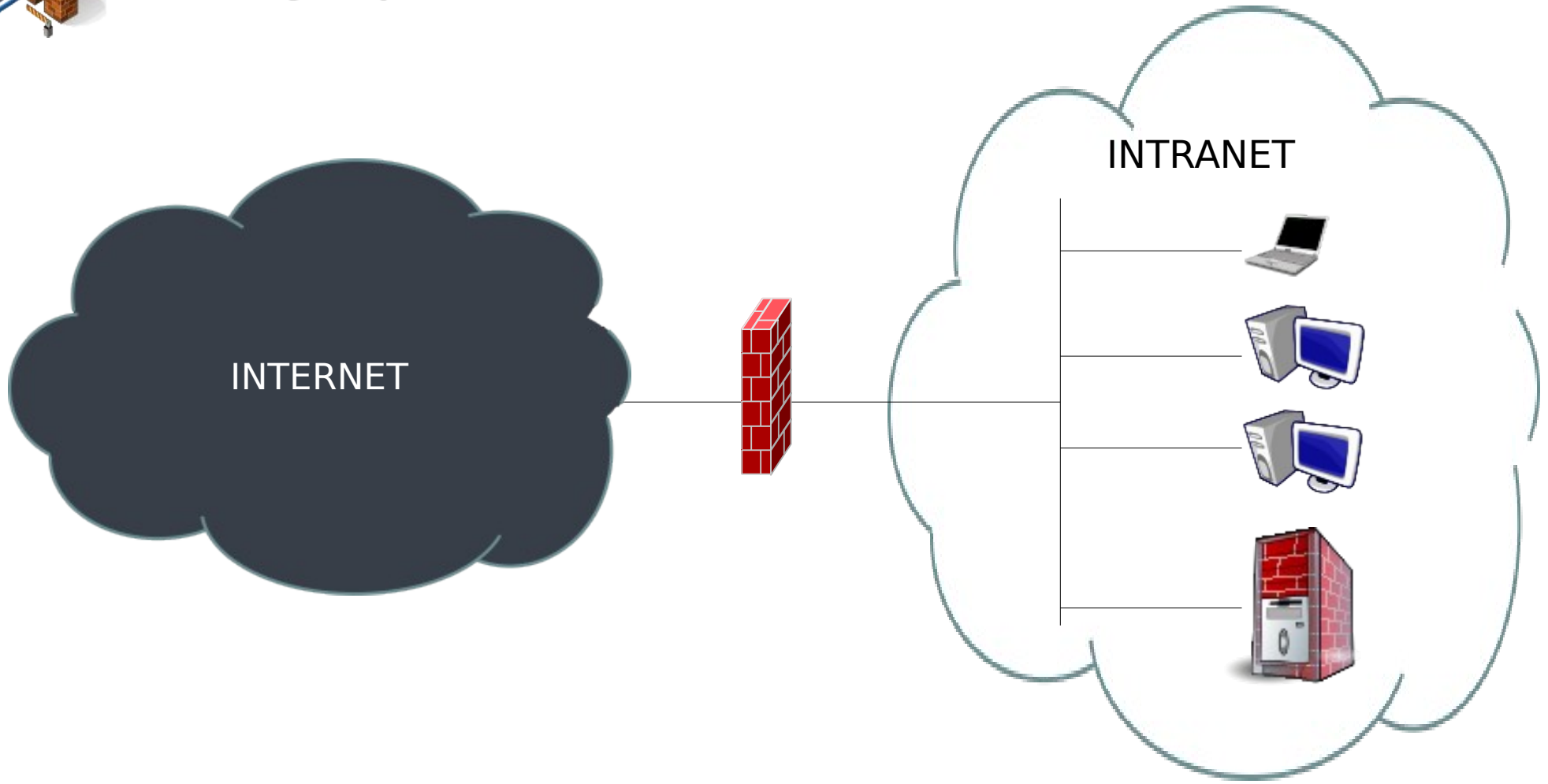
```
[root@localhost:~]# iptables -A INPUT -m state --state \
                    ESTABLISHED,RELATED -j ACCEPT
[root@localhost:~]# iptables -A INPUT -p tcp --dport 22 --syn -j ACCEPT
[root@localhost:~]# iptables -A INPUT –j DROP
```

```
[root@localhost:~]# iptables-save | grep '\-A INPUT' | nl
     1  -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
     2  -A INPUT -p tcp -m tcp --dport --tcp-flags FIN,SYN,RST,ACK SYN -j ACCEPT
     3  -A INPUT -j DROP
```

```
[root@localhost:~]# iptables -nvL
Chain INPUT (0 references)
 pkts bytes target prot opt in out source    destination
    0     0 ACCEPT all  --  *  *   0.0.0.0/0 0.0.0.0/0    state RELATED,ESTABLISHED
    0     0 ACCEPT tcp  --  *  *   0.0.0.0/0 0.0.0.0/0    tcp dpt:22 flags:0x17/0x02
    0     0 DROP   all  --  *  *   0.0.0.0/0 0.0.0.0/0
```

Firewalling and monitoring

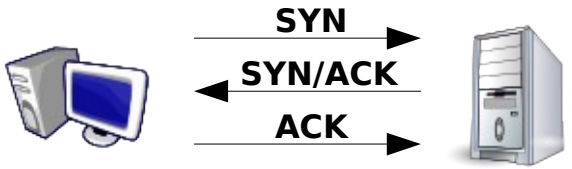
Firewall?



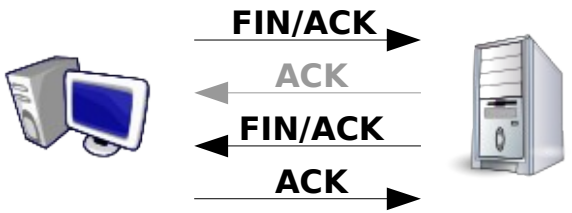
TCP Connection

TCP CONNECTION TOWARD AN OPEN PORT (listening)

CLIENT SERVER



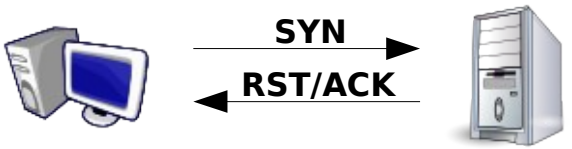
3-Way handshake
- SYN/half-open scan
- connect()

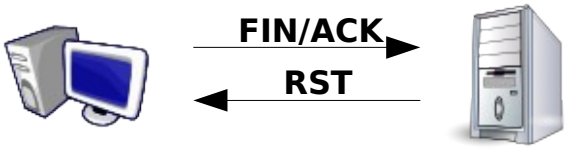


normal ending
(if ESTABLISHED)

TCP CONNECTION TOWARD A CLOSE PORT (non-listener)

CLIENT SERVER





RFC 793

UDP & ICMP

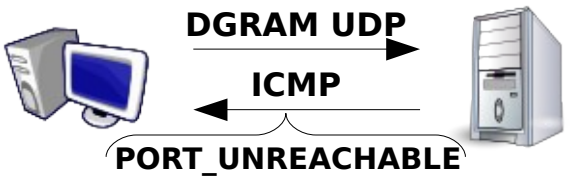
UDP CONNECTION TOWARD AN OPEN PORT (listening)

CLIENT SERVER

Depending on the application, there could be a reply or not

UDP CONNECTION TOWARD A CLOSED PORT (non-listener)

CLIENT SERVER



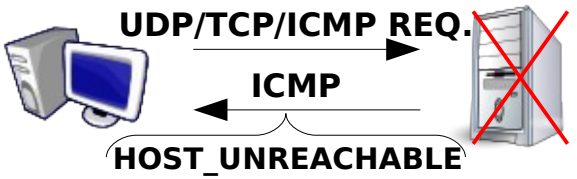
ICMP REQUEST/REPLY

CLIENT SERVER

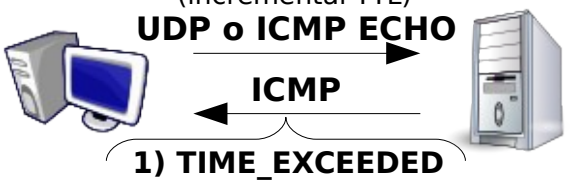
PING

ICMP NOTIFY

CLIENT SERVER



TRACEROUTE (*)
(incremental TTL)



2) PORT_UNREACHABLE / ECHO REPLY

RFC 768
RFC 792

(*) TCPTRACEOUTE does the same but uses TCP SYN, SYN/ECN/CWR or ACK and expects a SYN/ACK or RST, RST/ACK

Firewall verification tools

ACTIVE CHECK: port-scanner, diagnostic utilities, applications. The aim is to simulate an attack (port-scan, ping-sweep, ...) or simply triggering the firewall rules.

- nmap, hping, netcat, nessus
- ping, traceroute, tcptraceroute
- telnet, client ssh, browser, ...

PASSIVE CHECK: sniffing, diagnostic, logging. The aim is to check the traffic in order to verify the firewall response.

- tcpdump, wireshark/ethereal, iptraf, ettercap, dsniff
- iptstate, ss
- lsof, netstat, fuser
- syslogd, ulogd

Verify: TCP/UDP

Some examples to verify the rules concerning TCP and UDP protocols:

- using **nmap**

```
nmap [-sS|-sA|-sN|-sF|-sX|-sT] -p 21,22,23,25,80,443 -P0 ADDRESS
nmap -sU -sS -p T:21-23,25,80,443,U:53,123 ADDRESS -g 80
```

- using **hping**

```
hping -c 1 [-S,-A,-F,-R,-U,-P,-X,-Y] -p 80 -V ADDRESS
hping --udp -c 1 -p 123 -V ADDRESS -s 123 [--spoof NTPSERVER_ADDRESS]
hping -A --scan 22,80,443 -V ADDRESS -s 80
```

- using **nc (netcat)**

```
nc -z -vv ADDRESS 21 22 23 25 80 443
nc -z -vv -u ADDRESS 53 123
nc -v -u ADDRESS 53
```

  open a listening socket:

```
echo ciao | nc -v -l -u -p 53 LISTENING_ADDRESS
nc -v -l -p 80 LISTENING_ADDRESS
```

- using **telnet**

```
telnet ADDRESS PORT
```

- using (**tcp**)**traceroute**

```
(UDP) traceroute ADDRESS
(TCP) tcptraceroute [-S|-A] ADDRESS [PORT]
```

Verify: ICMP

Some examples to verify the firewall rules concerning ICMP:
(see /usr/include/linux/icmp.h for details about ICMP types and codes)

- *echo request* (ping)
  ```
  ping ADDRESS
  traceroute -I ADDRESS
  ```
- ping flood
  ```
  ping -f ADDRESS
  hping --icmp --icmptype 8 -c 100 --faster ADDRESS
  ```
- *timestamp request*
  ```
  hping --icmp --icmptype 13 -c 2 ADDRESS
  ```
- *destination unreachable - host unreachable*
  ```
  hping --icmp --icmptype 3 --icmpcode 1 -c 1 ADDRESS
  ```
- *destination unreachable - administratively prohibited*
  ```
  hping --icmp --icmptype 3 --icmpcode 13 -c 1 ADDRESS
  ```
- oversized ICMP
  ```
  ping -c 1 -s 65507 ADDRESS
  hping --icmp --icmptype 8 -d 65000 -c 1 ADDRESS
  ```

Verify: monitoring

Some commands to verify the connections status and network traffic:

- iptables counters and ACLs:

```
iptables -nvL [CHAIN]
iptables-save
```

- *connection tracking* status:

```
iptstate [-1]
```

- connessions and resources:

```
lsof -i TCP[:PORTA] -n -P
netstat -putan
fuser -v -n tcp 22 25 80 443
ss -4 -n -l
```

- traffic sniffing:

```
tcpdump -i INTERFACE -nn [-v]    \
     [proto PROTOCOL and port PORT and host ADDRESS ...]
{tshark|tethereal} -i INTERFACE -np [-V]    \
     [proto PROTOCOL and port PORT and host ADDRESS ...]
{wireshark|ethereal} &
iptraf -i INTERFACE
```

- kernel log checking (iptables -j LOG):

```
dmesg
tail -f /var/log/iptables.log  (by configuring syslogd properly)
```

That's All Folks!

(questions ; comments) | mail -s uheilaaa baro@democritos.it

(complaints ; insults) &>/dev/null

REFERENCES AND USEFUL LINKS

SOFTWARE:

- Linux Kernel http://www.kernel.org
- Netfilter http://www.netfilter.org

- nmap http://www.insecure.org/nmap/
- hping http://www.hping.org/
- netcat http://netcat.sourceforge.net/
- iptstate http://www.phildev.net/iptstate/
- ss http://linux-net.osdl.org/index.php/Iproute2
- lsof ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof/
- netstat http://www.tazenda.demon.co.uk/phil/net-tools/
- tcpdump http://www.tcpdump.org
- wireshark http://www.wireshark.org
- ethereal http://www.ethereal.com (vedi wireshark)
- iptraf http://iptraf.seul.org/
- ettercap http://ettercap.sourceforge.net
- dsniff http://www.monkey.org/~dugsong/dsniff/
- tcptraceroute http://michael.toren.net/code/tcptraceroute/
- (telnet, traceroute, ping, ...)

DOC:

- IPTables HOWTO http://www.netfilter.org/documentation/HOWTO/
- IPTables tutorial http://iptables-tutorial.frozentux.net/
- Having fun with IPTables
  http://www.ex-parrot.com/~pete/upside-down-ternet.html
- Denial of Service http://www.cert.org/tech_tips/denial_of_service.html
- IPv4 Address space
  - http://www.cymru.com/Documents/bogon-bn.html
  - http://www.iana.org/assignments/ipv4-address-space
  - http://www.oav.net/mirrors/cidr.html
  - http://en.wikipedia.org/wiki/IPv4
  - IANA http://www.iana.org
  - RIPE http://www.ripe.net
  - RFC 3330 http://www.rfc.net/rfc3330.html
- SANS: http://www.sans.org/reading_room/whitepapers/firewalls/
  http://www.sans.org/reading_room/

RFC: (http://www.rfc.net)

- RFC 791 – Internet Protocol (IPv4)
  http://www.rfc.net/rfc791.html
- RFC 793 – Transmission Control Protocol (TCP)
  http://www.rfc.net/rfc793.html
- RFC 768 – User Datagram Protocol (UDP)
  http://www.rfc.net/rfc768.html
- RFC 792 – Internet Control Message Protocol (ICMP)
  http://www.rfc.net/rfc792.html
- RFC 1180 – A TCP/IP Tutorial
  http://www.rfc.net/rfc1180.html
- RFC 1700 / IANA db – Assigned Numbers
  http://www.rfc.net/rfc1700.html
  http://www.iana.org/numbers.html
- RFC 3330 – Special-Use IPv4 Addresses
  http://www.rfc.net/rfc3330.html
- RFC 1918 – Address Allocation for Private Internets
  http://www.rfc.net/rfc1918.html
- RFC 2196 – Site Security Handbook
  http://www.rfc.net/rfc2196.html
- RFC 2827 – Network Ingress Filtering
  http://www.rfc.net/rfc2827.html
- RFC 2828 – Internet Security Glossary
  http://www.rfc.net/rfc2828.html
- RFC 1149 – Transmission of IP Datagrams on Avian Carriers
  http://www.rfc.net/rfc1149.html
- Unofficial CPIP WG
  http://www.blug.linux.no/rfc1149/
- RFC 2549 – IP over Avian Carriers with Quality of Service
  http://www.rfc.net/rfc2549.html
- Firewalling the CPIP
  http://www.tibonia.net/
  http://www.hotink.com/wacky/dastrdly/

Some acronyms...

IP – Internet Protocol
TCP – Transmission Control Protocol
UDP – User Datagram Protocol
ICMP – Internet Control Message Protocol
ARP – Address Resolution Protocol
MAC – Media Access Control

OS – Operating System
NOS – Network Operating System
LINUX – LINUX is not UNIX

PING – Packet Internet Groper

FTP – File Transfer Protocol – (TCP/21,20)
SSH – Secure SHell – (TCP/22)
TELNET – Telnet – (TCP/23)
SMTP – Simple Mail Transfer Protocol – (TCP/25)
DNS – Domain Name System – (UDP/53)
NTP – Network Time Protocol – (UDP/123)
BOOTPS – Bootstrap Protocol Server (**DHCP**) – (UDP/67)
BOOTPC – Bootstrap Protocol Server (**DHCP**) – (UDP/68)
TFTP – Trivial File Transfer Protocol – (UDP/69)
HTTP – HyperText Transfer Protocol – (TCP/80)
NTP – Network Time Protocol – (UDP/123)
SNMP – Simple Network Management Protocol – (UDP/161)
HTTPS – HyperText Transfer Protocol over TLS/SSL – (TCP/443)
RSH – Remote Shell – (TCP/514,544)

ISO – International Organization for Standardization
OSI – Open System Interconnection

TLS – Transport Layer Security
SSL – Secure Sockets Layer

RFC – Request For Comments

ACL – Access Control List

PDU – Protocol Data Unit

TCP flags:
- **URG**: Urgent Pointer field significant
- **ACK**: Acknowledgment field significant
- **PSH**: Push Function
- **RST**: Reset the connection
- **SYN**: Synchronize sequence numbers
- **FIN**: No more data from sender

RFC 3168 TCP flags:
- **ECN**: Explicit Congestion Notification
- (**ECE**: ECN Echo)
- **CWR**: Congestion Window Reduced

ISN – Initial Sequence Number