Lesson #6
Network management, security practices, and troubleshooting.

Advanced Linux Administration

Robin Hack

Introduction

30 or 45 min

Network scenario - too many questions

Network scenario

- How is server configured?
- What running on my server and can possible be reachable from outside world?

Basic toolset

5 min

Basics - toolset for Linux 1/2 - Common tools

`ping` - sends ICMP echo-request. Are remote end reachable? Also can be used to detect packet loss

```
# ping 8.8.8.8
```

`traceroute` - search way with help of UDP, TCP, ICMP - can be used to detect packet loss

```
# traceroute 8.8.8.8
```

`host` - DNS queries

```
# host microsoft.com
```

`tcpdump` - record and analyse content of datagrams (CLI)
`WireShark` - record and analyse content of datagrams (GUI)

`nc` - It can open TCP connections, send UDP packets, listen on arbitrary TCP and UDP ports, do port scanning, and deal with both IPv4 and IPv6.

```
$ nc -l 10.20.30.40 2222
```

Basics - toolset for Linux 2/2

Deprecated tools: `ifconfig`, `route`, `netstat`, `brctl`

New tools (iproute2): `ip`, `ip route`, `ss`, `ip link`

`ifconfig` (deprecated) vs `ip` - configure/discovery network interfaces/addresses

Configuration - non persistent changes. Everything will be lost after reboot.

Privileged operations. You need to be superuser.

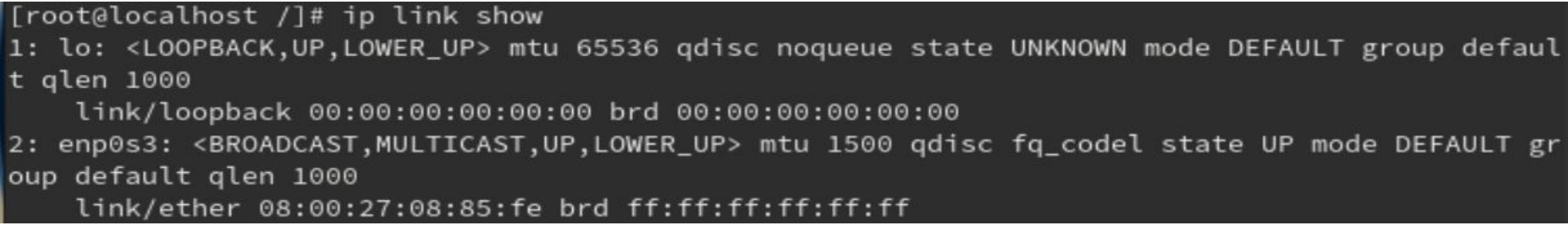
Examine network configuration

10 min

Examine network configuration - interfaces

Check interfaces in my system:

```
# ip link show
```



```
[root@localhost /]# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT gr
oup default qlen 1000
    link/ether 08:00:27:08:85:fe brd ff:ff:ff:ff:ff:ff
```

Virtual interfaces vs physical interfaces.

`1: lo` -> virtual network interface that your computer uses to communicate with itself. It is used mainly for diagnostics and troubleshooting, and to connect to servers running on the local machine.

`2: enp0s3` -> physical interface

Virtual network interface - loopback aka lo

The loopback device is a special, virtual network interface that your computer uses to communicate with itself.

It means, that even computer without any real hardware ethernet adapter can start services which uses (not only) TCP/IP protocol.

You can test/develop your server service in safe environment.

Virtual network interface - dummy

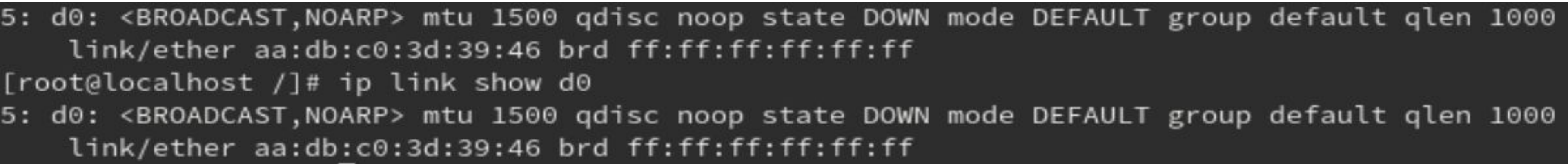
Why? Testing and debugging.

Add interface:

```
# ip link add d0 type dummy up
```

Check interface:

```
# ip link show d0
```

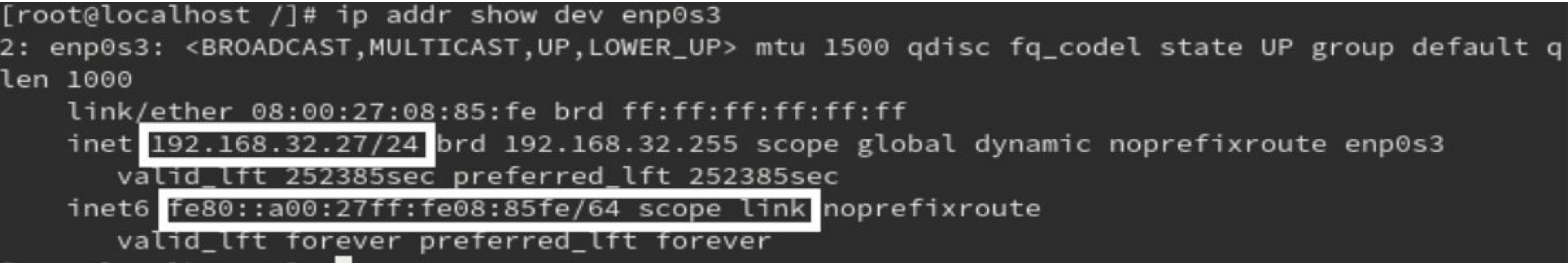


```
5: d0: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether aa:db:c0:3d:39:46 brd ff:ff:ff:ff:ff:ff
[root@localhost /]# ip link show d0
5: d0: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether aa:db:c0:3d:39:46 brd ff:ff:ff:ff:ff:ff
```

NOTE: All datagrams of dummy interface go through loopback device (you will need at WireShark part).

Examine network configuration - addresses

```
# ip address show (optionally: dev name_of_interface)
```



```
[root@localhost /]# ip addr show dev enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default q
len 1000
    link/ether 08:00:27:08:85:fe brd ff:ff:ff:ff:ff:ff
    inet 192.168.32.27/24 brd 192.168.32.255 scope global dynamic noprefixroute enp0s3
       valid_lft 252385sec preferred_lft 252385sec
    inet6 fe80::a00:27ff:fe08:85fe/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

inet -> address family: ipv4

192.168.32.26/24 -> ipv4 address/mask

DEMO - Configure virtual dummy network interface

Also applicable to real interfaces.

Configure:

```
# ip link add d0 type dummy
# ip addr add dev d0 10.20.30.40
# ip link set dev d0 up
```

Check:

```
# ip a s d0
# ping 10.20.30.40
```

What can be (possibly) reachable from network?

Old (deprecated):

- `# netstat -ltnp`

New:

- `# ss -ltnpZ`

`-l` - listen; `-t` - TCP; `-n` - don't resolve hostnames; `-p` - process; `-Z` - SELINUX security contexts

Deprecated Linux networking commands and their replacements

What can be (possibly) reachable from network?

Example output of `ss -ltnp`:

```
State  Recv-Q Send-Q  Local Address:Port Peer Address:Port
LISTEN 0      128     *:22                *:*           users:(("sshd",pid=978,fd=3))
```

State: LISTEN -> able to accept connection from clients

Local Address: `*:22` means listen to all addresses on port TCP 22

users(...): name of process, pid of process

What can be (possibly) reachable from network?

Why possibly reachable?.

What can be (possibly) reachable from network?

Why possibly reachable?

Firewall can block incoming datagrams.

Firewall on Linux

10 min

Basics - Firewall on Linux

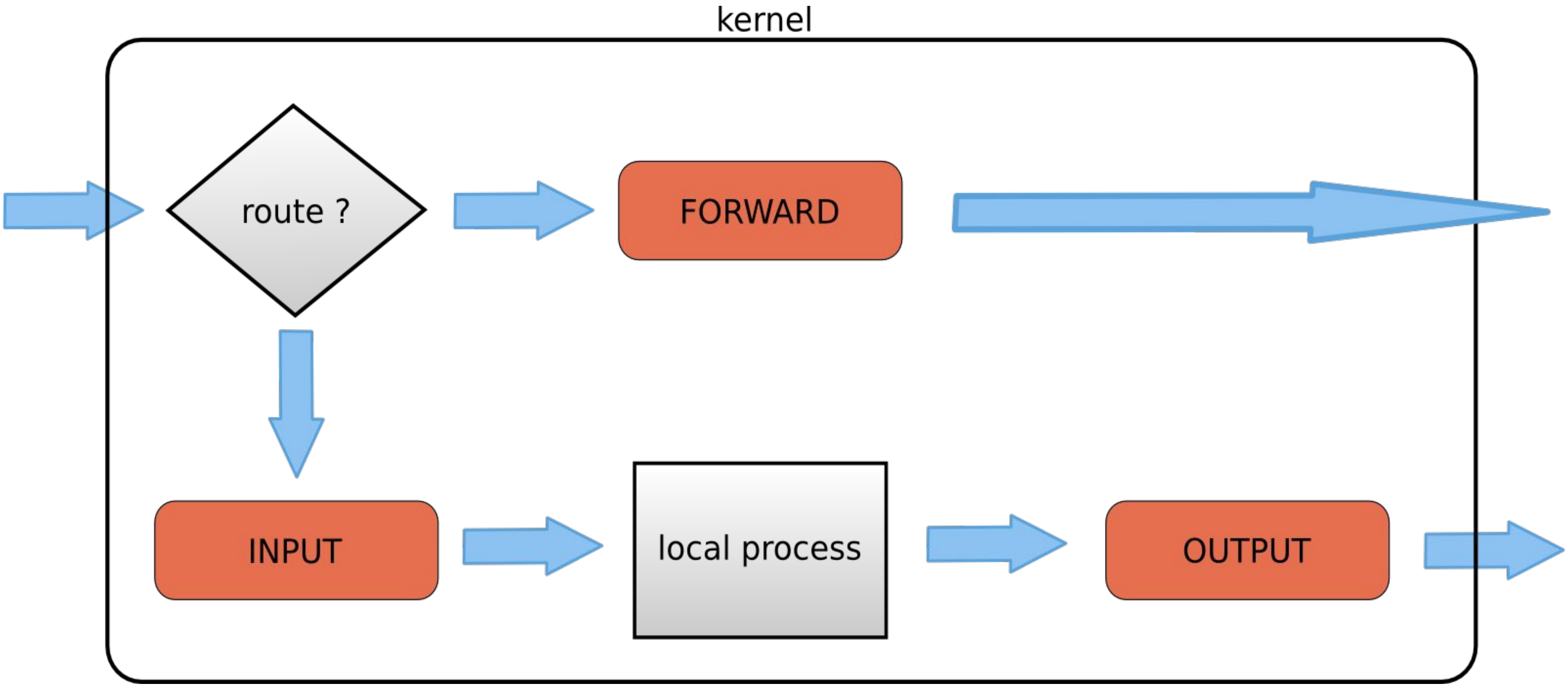
Once again, deprecated tools vs new tools.

Old tools (STILL USED): `{ip,ip6,arp,eb}tables`

New tools: nftables (command: `nft`) is a netfilter project that aims to replace the existing `{ip,ip6,arp,eb}tables` framework. It provides a new packet filtering framework, a new user-space utility (nft).
Will be used in future. Totally different approach.

On Fedora: `firewalld` (build on top of iptables/nft) - we will not use it here now

Basics - Firewall on Linux - iptables



Basics - Firewall on Linux - iptables

- Policies - ACCEPT, DROP
- Chains - INPUT, OUTPUT, FORWARD
- Targets - ACCEPT, DROP, REJECT

Basics - Firewall on Linux - iptables

When no rule is applied, then default policy is used.

Policy:

ACCEPT - accept datagrams

DROP - drop datagrams

Example:

```
# iptables -P INPUT DROP
```

Congratulation! You lost connection to server and you need to reach it physically (or use remote KVM)

Basics - Firewall on Linux - iptables

Chains:

INPUT - incoming datagrams

OUTPUT - outgoing datagrams

FORWARD - routed datagrams

Rules are stored in chains

Each chain have default policy: `ACCEPT`

Policy defines what to do with datagram when no rules were applied.

Basics - Firewall on Linux - iptables

Targets:

REJECT - politely say: no, to datagrams

DROP - just blackhole for datagrams

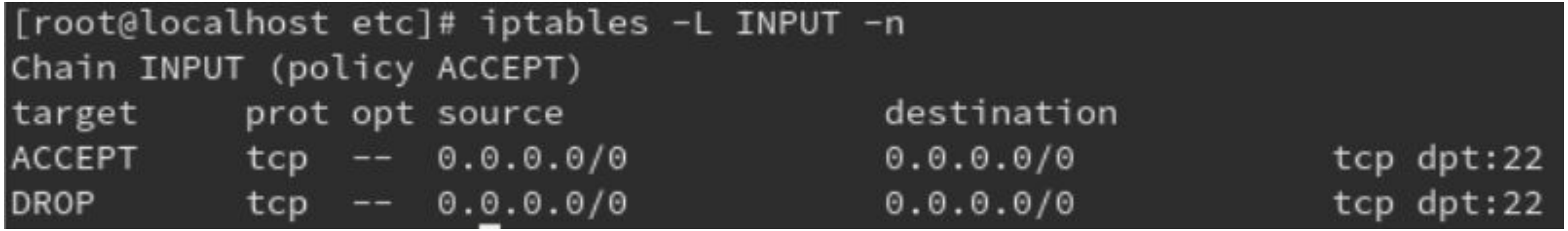
ACCEPT - allow datagrams

Basics - Firewall on Linux - iptables - rules and rules order

From top to bottom.

First match wins!

Example: tcp datagram with destination port 22



```
[root@localhost etc]# iptables -L INPUT -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:22
DROP       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:22
```

Basics - Firewall on Linux - iptables - Adding rules

Basic anatomy of firewall rule:

```
# iptables -A "chain name" -i "interface" -p "protocol" -s "source address" -d
"destination address" --dport "destination port" -j "target"
```

Chain name: INPUT, OUTPUT, FORWARD

Interface: d0, ens30p, ... (optional)

Protocol: tcp, udp, icmp, ...

Destination address: 10.20.30.40 (optional)

Source address: 192.168.0.10 (optional)

Destination port: number or name (22 or ssh)

Target: REJECT, DROP, ACCEPT

Destination port for services?! You can take a look at `/etc/services`

Basics - Firewall on Linux - iptables - recapitulation

Show firewall rules:

```
# iptables -L -n
```

Add firewall rule:

```
# iptables -A INPUT -i d0 -p tcp --dport 2222 -j DROP
```

Remove firewall rule:

```
# iptables -D INPUT -i d0 -p tcp --dport 2222 -j DROP
```

Remove (aka flush) all rules:

`# iptables -F` (optionally: chain name)

Set default policy for INPUT chain:

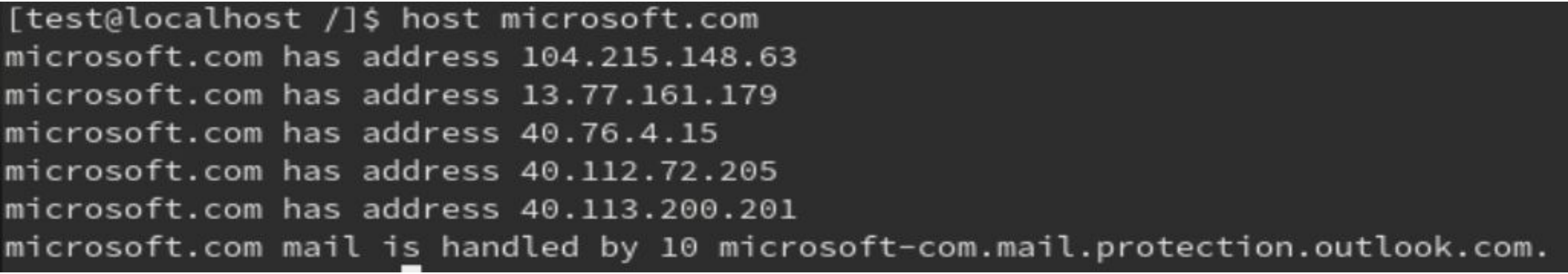
```
# iptables -P INPUT ACCEPT
```

DNS

5 min

DNS - Domain name system

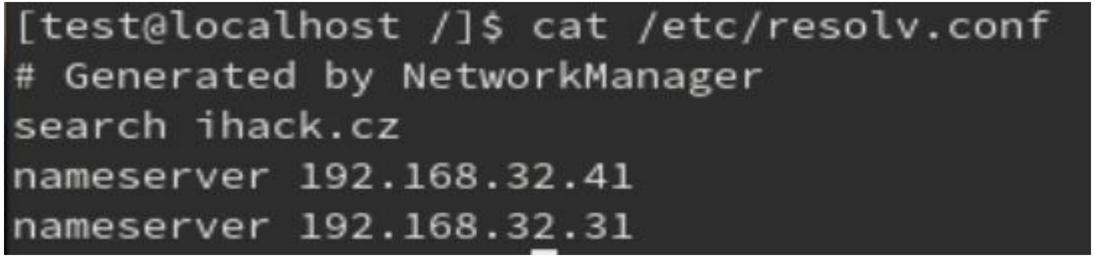
DNS translate human readable form to machine readable form.



```
[test@localhost /]$ host microsoft.com
microsoft.com has address 104.215.148.63
microsoft.com has address 13.77.161.179
microsoft.com has address 40.76.4.15
microsoft.com has address 40.112.72.205
microsoft.com has address 40.113.200.201
microsoft.com mail is handled by 10 microsoft-com.mail.protection.outlook.com.
```

DNS - configuration

`/etc/resolv.conf`



```
[test@localhost /]$ cat /etc/resolv.conf
# Generated by NetworkManager
search ihack.cz
nameserver 192.168.32.41
nameserver 192.168.32.31
```

`nameserver` -> server where we send requests to (resolver)

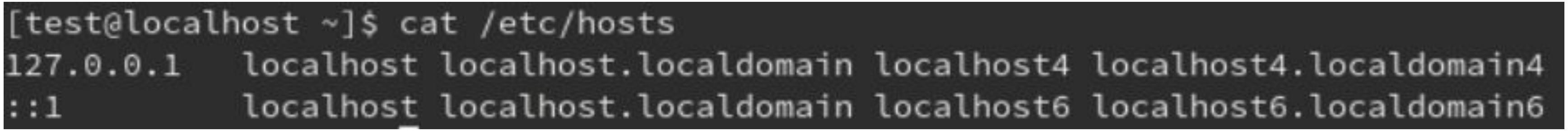
Public resolvers:

8.8.8.8 - Google

1.1.1.1 - Cloudflare

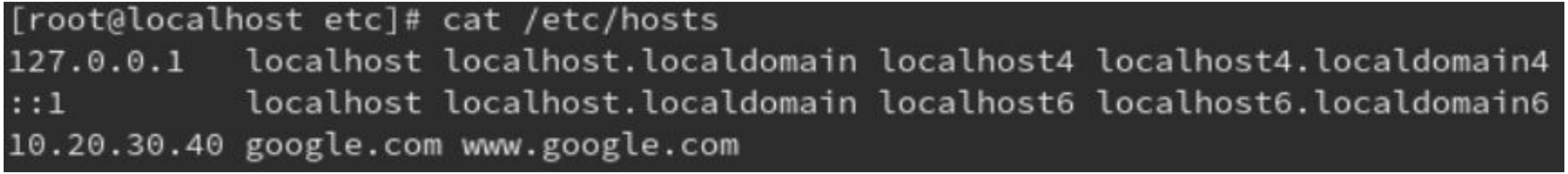
DNS - configuration - /etc/hosts

Hosts were used in dawn of internet instead of DNS. They must be copied from source to all machines in Internet.



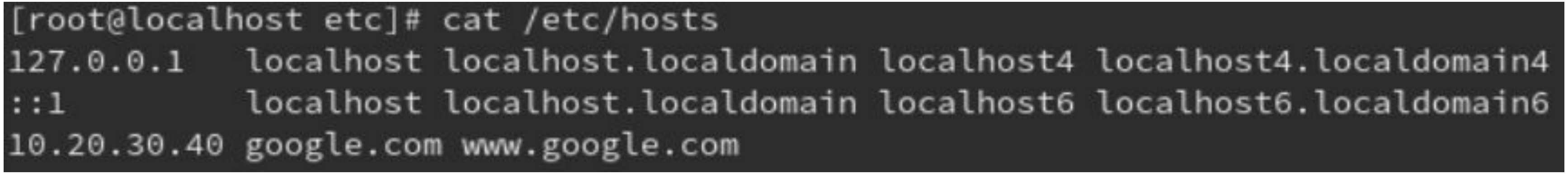
```
[test@localhost ~]$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
```

Edited:



```
[root@localhost etc]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.20.30.40 google.com www.google.com
```

DNS - configuration - /etc/hosts - and system



```
[root@localhost etc]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.20.30.40 google.com www.google.com
```

```
$ host www.google.com
www.google.com has address 216.58.201.68

$ ping www.google.com
PING google.com (10.20.30.40) 56(84) bytes of data.
```

DNS - configuration - /etc/hosts - and system

```
$ host www.google.com
```

asks directly to DNS resolvers which are stored in file `/etc/resolv.conf`

```
$ ping www.google.com
```

asks `/etc/hosts` first then DNS resolvers

DNS - configuration - troubleshooting

```
# host microsoft.com
```

took so long to resolve...

Possible issue:
one resolver from `/etc/resolv.conf` is down

```
# ping 192.168.32.41
```

No response to ICMP echo -> server is down or connection to server is broken.

Default timeout is 5 seconds.

Netcat tool

5 min

Netcat aka: nc, ncat, ...

Why you should care?

It is a feature-rich network debugging and investigation tool.

Netcat aka: nc, ncat, ...

Two modes:

Listen aka server mode:

```
$ nc -l -c "echo 'hello'" 10.20.30.40 2222
$ ss -ltp | grep 2222
```

Client mode

```
$ nc 10.20.30.40 2222
hello
```

Possible issues - not only netcat (demonstration only)

First terminal:

```
$ nc -l 10.20.30.40 1234
```

Second terminal:

```
$ nc -l 10.20.30.40 1234
Ncat: bind to 10.20.30.40:1234: Address already in use. QUITTING.
```

Port can be used only by one process.

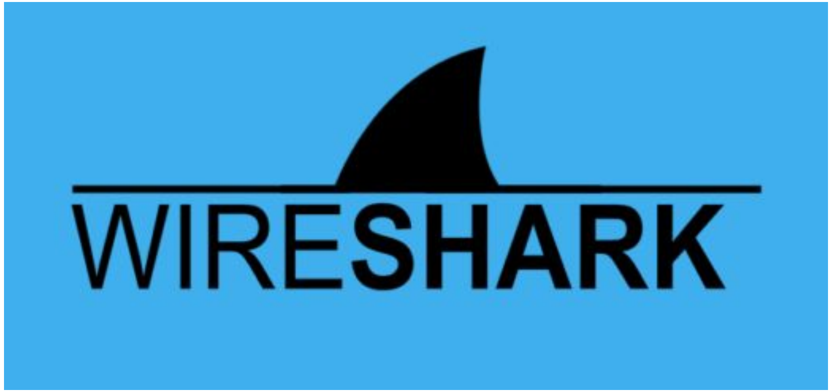
Possible issues - not only netcat (demonstration only)

```
$ nc -l 10.20.30.40 1023
Ncat: bind to 10.20.30.40:1023: Permission denied. QUITTING
```

Well Known Port Numbers (0 to 1023)
Can be used only by root user.
Examples:

80 - http
443 - https

Why restricted? Without restriction, not only evil user can start http server.



Wireshark tool

5 min

Wireshark

Prerequisites (all commands under root):

```
# dnf install wireshark
# usermod -G wireshark -a YOUR_USERNAME
# chmod o+x /usr/bin/dumpcap
```

Logout from running session.

Wireshark

DEMO

Security?!

Someone on network can read your data. What it means?
Security is not only about firewall. It's even about transmitted data.

Use encrypted protocols.

```
http           -> https
ftp            -> ftps
telnet, rsh    -> ssh
rcp            -> scp, sftp
smtp           -> smtps
```

Workshop

60 min

Workshop labs

- In the following next slides there are 5 labs total
- Each lab has a time estimate how much time should you spend on it if everything goes well
- We encourage you to help each other or rise your hand to get help from lecturers
- If you couldn’t finish all labs during this class, you should complete them on your own later because learned skills will be used in following lectures or during a final practical exam.
- *NOTE: Focus on the lab content and leave any exploration or deep dive desires as a self-study for later.*

Workshop labs - before you begin

Prerequisite for all labs:

```
# systemctl stop firewalld
```

Workshop labs - before you begin

- Some changes can be hold back. Only reboot helps.
- Changes are not persistent. They will not survive reboot.

Useful help:

Set back default policies:

```
# iptables -P INPUT ACCEPT
```

Flush firewall settings:

```
# iptables -F
```

Labs

1. Network interfaces and Wireshark (10min)
    a. dummy interfaces, install/setup wireshark, netcat
2. Basic iptables configuration (10 min)
    a. deny all by default, allow client-server with netcat, review iptables rules, allow ssh rule
3. SSHD configuration and setup (10min)
    a. Install, allow ssh service via iptables, copy file
4. Broken DNS (10 min)
    a. set resolver to wrong state, exercise /etc/hosts /etc/resolv.conf
5. Detecting packet loss (5 min)
    a. use ping and traceroute tool to detect packet loss

Lab 1 - Network interfaces and Wireshark (10 min)

Lab 1 - What you will learn?

- How to setup and use Wireshark
- How to setup dummy interfaces
- How to use netcat tool in listen and client mode
- How to breach privacy of traffic

Lab 1 - Intro

- install Wireshark
- setup Wireshark that can be used under your user account
- create dummy network interface `d0` with address 10.20.30.40
- start netcat TCP service on port 79 listen only on address 10.20.30.40 and check that service is running
- start Wireshark and listen on `Loopback:lo` interface
- connect to netcat TCP service on port 79 and send "magic secret string"
- find "magic secret string" in Wireshark

Lab 1 - Guided solution

- install Wireshark
  ```
  # dnf install wireshark
  ```
- setup Wireshark that can be used under your user account
  ```
  # dnf install wireshark
  # usermod -G wireshark -a YOUR_USERNAME
  # chmod o+x /usr/bin/dumpcap
  ```
- create dummy network interface `d0` with address `10.20.30.40`
  ```
  # ip link add d0 type dummy d0
  # ip addr add dev d0 10.20.30.40
  # ip link set d0 up
  ```
- start netcat TCP service on port `79` listen only on address `10.20.30.40` (you will need to start service under root)
  ```
  $ sudo nc -l 10.20.30.40 79
  # ss -ltnp
  ```
- start Wireshark and listen on `Loopback - lo` interface
- connect to netcat TCP service on port 79 and send "magic secret string"
  ```
  $ echo "magic secret string" | nc 10.20.30.40 79
  ```
- find "magic secret string" in Wireshark

Lab 1 - Cleanup

```
# killall nc

# ip addr del dev d0  10.20.30.40/32

# ip link del dev d0
```

Lab 2 - Basic iptables configuration (10 min)

Lab 2 - What you will learn?

Main goal: Learn how to use firewall on Linux via `iptables` with default policies.

- Start and stop netcat TCP service
- Add rule(s) to firewall
- Check rules
- Set default policy on firewall
- Check firewall configuration

Lab 2 - Intro

- Start netcat TCP server listening on port 2222 and check that service running
- Start netcat TCP server on port 4444 and check that service running
- Allow netcat TCP server on port 2222 for whole world on firewall
- Allow ssh service via TCP protocol for whole world on firewall
- Default policy drops all incoming datagrams
- Check firewall rules
- Check that netcat TCP server on port 2222 is reachable
- Check that netcat TCP server on port 4444 is NOT reachable

Lab 2 - Guided solution

- Start netcat TCP service port 2222
  ```
  $ while :; do echo "2222" | nc -l 2222; done
  $ ss -ltnp | grep 2222
  ```
- Start netcat TCP service on port 4444
  ```
  $ while :; do echo "4444" | nc -l 4444; done
  $ ss -ltnp | grep 4444
  ```
- Allow netcat TCP service on port 2222 for whole world on firewall
  ```
  # iptables -A INPUT -p tcp --dport 2222 -j ACCEPT
  # iptables -A INPUT -p tcp --sport 2222 -j ACCEPT
  ```
- Allow ssh service via TCP protocol for whole world on firewall
  ```
  # iptables -A INPUT -p tcp --dport 22 -j ACCEPT
  # iptables -A INPUT -p tcp --sport 22 -j ACCEPT
  ```
- Default policy drops all incoming datagrams
  ```
  # iptables -P INPUT DROP
  ```
- Check firewall rules
  ```
  # iptables -L
  ```
- Check that netcat TCP service on port 2222 is reachable
  `$ nc your_ip_address 2222` -> you should see 2222 on terminal
- Check that netcat TCP service on port 4444 is NOT reachable
  `$ nc your_ip_address 4444` -> should timeout, nothing on terminal

Lab 2 - Cleanup

Don't forgot change default policy back to ACCEPT and to flush your firewall rules.

```
# killall nc

# iptables -P INPUT ACCEPT

# iptables -F INPUT
```

Lab 3 - SSHD configuration and setup (5 min)

Lab 3 - Intro

- Install sshd service (`openssh-server`)
- Set firewall to allow sshd service (tcp port 22)
- Set default policy to drop all other services
- Check that ssh connection from your hypervisor still works

Lab 3 - Guided solution

- install sshd service (`openssh-server`)
  ```
  # dnf install openssh-server
  # systemctl start sshd
  ```
- set firewall to allow sshd service (tcp port 22)
  ```
  # iptables -A INPUT -p tcp --dport 22 -j ACCEPT
   # iptables -A INPUT -p tcp --sport 22 -j ACCEPT
  ```
- set default policy to drop all other services
  ```
  # iptables -P INPUT DROP
  ```
- check that ssh connection from your hypervisor still works
  On your virtual machine:
  `$ ip addr show` -> and find usable ip
  From your hypervisor:
  ```
  $ scp /etc/passwd user@ip_of_your_virtual_machine:/tmp/
  ```

Lab 3 - Cleanup

Don't forgot change default policy back to ACCEPT and to flush your firewall rules.

```
# iptables -P INPUT ACCEPT

# iptables -F INPUT
```

Lab 4 - Broken DNS
(5 min)

Lab 4 - Intro

- Resolve address of domain redhat.com.
- Edit `/etc/resolv.conf` to use resolvers:

  `10.20.30.40` - first

  `8.8.8.8` - second
- Resolve address of redhat.com again
  - Is it slower than before?
- Add redhat.com with address `10.20.30.40` to `/etc/hosts`
- Try to resolve redhat.com with `ping` and with `host`
- Try to answer some questions:
  - Is here difference at resolved addresses?
  - Is here some difference in times?

Lab 4 - Cleanup

Return `/etc/resolv.conf` to previous state

Remove `redhat.com` from `/etc/hosts` file

Lab 5 - Detecting packet loss (5 min)

Lab 5 - Intro

- Try to detect packet loss to site `redhat.com` with `netstat` and `ping` tool

Lab 5 - Guided solution

Try to detect packet loss with netstat and ping tool

Ping tool:

```
$ ping -c 10 redhat.com
```

Possible output:

```
10 packets transmitted, 10 packets received, 0.0% packet loss
```

Traceroute tool:

```
$ traceroute -I redhat.com
```

Possible output:

```
* * redhat.com (209.132.183.105)  175.083 ms
```

"*" asterisk means that there should be packet loss

Links and Resources

LinuxDays 2019 - Pokročilejší síťování v Linuxu - Ondřej Caletka

Zjednodusene zaklady prace s IPTABLES

How To Use Traceroute and MTR to Diagnose Network Issues