Linux System Administration

PM-TRTNB319

Zsolt Schäffer, PTE-MIK

Legal note

These slides form an outline of the curriculum and contain multiple references to the official Red Hat training materials (codes RH124, RH134 and RH254), since students of this subject are also eligible for the named RH learning materials. Some diagrams and pictures originating from the RH courses will show up in this outline as there is an agreement in place which allows for our Faculty to teach this subject based on the Red Hat curriculum.

All other pictures and diagrams sourced from third parties are explicitly marked with a reference to the origin.

Be aware that the mentioned parts and also this series of slides as a whole are property of their respective owners and are subject to copyright law.

Legal note

Network settings

Lecture 7

Chapter 1

Interfaces and networks

- There are several ways to automate networking setup at system startup.
- Legacy Sys V init systems have a network script (/etc/init.d/network) that sets up network interfaces at boot, based on files in /etc/sysconfig/network-scripts in Red Hat family, and files in /etc/network/interfaces in Debian family of Linux.
- Both families have the Network Manager daemon available that works according to configuration files in /etc/NetworkManager directory.
- The systemd system manager has a builtin network ma-nager called systemd-networkd, which works based on config files in the /etc/systemd/network directory.
- Only one of the above network manager daemons/ scripts should be enabled at a time in a system.

General network config files

- The /etc/hosts file serves as a basic name resolution service, it has name ↔ IP address pairs listed in text format. This overrides any other source of name resolution.
- The /etc/hostname file contains the hostname of the system. Use **hostnamectl status** and **hostnamectl set-hotname station.domain.tld** form to manage it.
- The /etc/resolv.conf contains the DNS resolver settings:

/etc/resolv.conf

```
search example.com company.net
domain test.org
#either use search or domain, not both
nameserver 192.168.0.100
nameserver 8.8.8.8
```

- The /etc/protocols, /etc/services files contain numeric ↔ alphabetical pairs of protocol and service identifiers.

Network config files

- The **/etc/sysconfig/network-scripts** directory contains the ifcfg, ifup and ifdown scripts. On a legacy RH system, they are the direct source of network settings.
- If the NM daemon is set up and enabled, it will create ifcfg files programatically.
- The NM daemon as well as systemd-networkd will manage it's own version of resolv.conf dynamically (based on DHCP replies and static connection settings). In this case the /etc/resolv.conf file is a symlink to the daemon-created instance (usually found somewhere in the /run directory).
- The systemd-resolved service has to be enabled separately from systemd-networkd for the above functionality to work. NetworkManager doesn't have this separated.

Interface naming

- A computer connects to networks via hardware devices called NICs (Network Interface Cards). A computer can have many interfaces, IFs can have multiple addresses.
- In legacy systems the interfaces are named by their type (physical layer mostly) and numbered in order of device node creation (udevd). E.g. eth0, eth1, wlan0, etc…
- In modern Linux systems the naming convention is the following:
  - The first two letters are assigned based on IF type, like for e.g. *en* or *wl*
  - The next letter is assigned based on IF connection, e.g. *p* for PCI, *o* for on-board, *s* for hot-plug interfaces
  - The last character is numeric, it represent port or index number.

Systemd networking

- You can choose an arbitrary name for the configuration files, but they should end with .network. Put them in /etc/systemd/network directory. Look up **https://www.freedesktop.org/software/systemd/man/systemd.network.html** for more details.

/etc/systemd/network/25-wireless.network

```
[Match]
Name=wlp2s0
[Network]
DHCP=ipv4
[DHCP]
RouteMetric=20
```

/etc/systemd/network/50-wired.network

```
[Match]
Name=enp1s0
[Network]
Address=10.1.10.9/24
Gateway=10.1.10.1
```

Querying networks and interfaces

- The legacy **ifconfig -a** command can be used to query all interfaces, you can also use **ifconfig *if-name addr*** command to set an address for a given interface. The ifconfig utility has limitations, it is considered obsolete.
- Use the **ip** utility to manage ip settings. Type **ip address show enp1** to see the address of the enp1 interface. Omitting IF name will result in printing all interface addresses.
- Issue **ip route show** to see the routing information.
- You can query IF statistics with **ip -s link show *if-name***.
- The **ss** command may be used to query listening and established connections of the processes running in the system. Note -t for TCP only, -u for UDP only, -l for listening sockests only, -p for printing assigned PID, -n for skipping host name and protocol name resolution.

Trouble-shooting networking

- Use **ping** to check connectivity. You can either specify an address or a hostname after ping. If the hostname is not resolved by ping, then your resolv.conf might not be in order. The -c switch can set a counter for ping.
- Use **host**, **nslookup** or **dig** for checking name resolution.

Shell command line

```
# nslookup -type=any google.com ns1.company.net
Server:         192.168.19.2
Address:   192.168.19.2#53
Non-authoritative answer:
Name:   google.com
Address: 173.194.35.7
Name:   google.com
Address: 173.194.35.8

google.com    nameserver = ns1.google.com. [...]
```

- Use **traceroute** or **tracepath** for checking routing path of a packet. (relies on ICMP, might be forbidden)

Network Manager concepts

- The NetworkManager daemon (NM) if enabled, is a persistent process, that automatically configures interfaces at boot, and handles all the interface events, like hot-plugging and interface or the loss/gain of a link/ carrier on interfaces.
- NM can be configured using a GUI, a TUI, and a CLI interface. It also has a Gnome notifier application for the graphical desktop. The command line interface will be discussed in detail in this lecture.
- NM can handle several objects, like interfaces, connections, radios, and general network settings.
- The most important objects for us are the device objects, which represent Linux network interfaces, and connection objects, which represent sets of settings applicable to interfaces.

Network Manager concepts

- A device can have many related connections but only one connection can actively use a device at a time.
- A device can be used in multiple networks, like for e.g. connecting the same computer's interface to a lab network for once that requires static config, then to a home network for once which provides DHCP services.
- Each connection can be saved by NM, and instead of changing each individual network setting manually, you can instruct NM to switch connections instead.
- Look at the output of **nmcli con show** and **nmcli dev status** !

nmcli

- NM's CLI interface is called **nmcli**. The general form of the command is as follows:
  nmcli [OPTIONS] OBJECT { COMMAND | help }
  where object (in our case) will either be device or connection. After that, you should specify a command like add, modify, etc… You can use the help command to get help relevant to the selected object.
- The words in the nmcli command line can be abbreviated to a non-ambigous level. For e.g **nmcli connection show**, **nmcli con sh**, **nmcli c s** all instruct for the same activity.
- You can also use TAB completion when dealing with nmcli commands.

nmcli con

- You can list all connections with **nmcli con show** or list only the active ones with **nmcli con show --active**.
- You can request all the details of a connection by naming it in the command: **nmcli con show "static-etho"**

Shell command line

```
# nmcli connection show
NAME        UUID              TYPE           DEVICE

docker0     33181c5b-[...]  bridge         docker0
dravanet    5dcea3ee-[...]  pppoe          enp3s0
enp2s0      41bba397-[...]  802-3-ethernet enp2s0
```

- You can list devices and their active connection with **nmcli dev status**. Get details by naming the device to the show subcommand, e.g. **nmcli dev show enp2so**.
- Refer to **RH124 CH 11.4.** for more.

nmcli con

- An example for adding connections: **nmcli con add con-name "default" type ethernet ifname eth1** This will set the connection to be a DHCP client since no address options have been specified. Another example: **nmcli con add con-name "demo" type ethernet ifname eth1 autoconnect no ip4 172.25.0.10/24 gw4 172.25.0.254** This will create a connection named demo for the eth1 interface with a staticIPv4 address.
- Issue **nmcli con add help** for all available property strings.
- You can switch between connections by issuing **nmcli con up "demo"** or **nmcli con up "default"**.
- You can administratively disable an interface with **nmcli disconnect dev eth1**. This will also terminate its current active connection.

nmcli con

- You can bring a connection down by **nmcli con down "demo"**. This will not disable the interface, meaning that in the event of a link change, the autoconnect connection for the device will be activated.
- You can modify an existing connection with **nmcli con mod *con-name property value*** like commands. E.g.: **nmcli con mod "static" connection.autoconnect yes**
- Some properties can have multiple values, like for e.g. ipv4.dns, since it is valid to have multiple resolvers configured in a network. You can assign and overwrite previous values the same way, e.g. **nmcli con mod "static" ipv4.dns 8.8.8.8** But in case of multi-value properties you can also prefix the setting with + or – to add/remove an entry, without modifying the other values of the same property. E.g. ... **+ipv4.dns 8.8.8.8**
- To delete a connection use **nmcli con del "demo"**.

Manual configuration of NM

- Complete practice **RH124 CH 11.5.** (nmcli)
- NM stores the connection properties in the /etc/sysconfig/network-scripts/ifcfg* files. This is directly compatible with RH network configuration schemes.
- You can also modify these files directly with an editor like vi. In this case you should notify NM of the changes. To achieve this, issue **nmcli con reload**. To apply the newly set values, issue **nmcli con down "con-name"** and **nmcli con up "con-name"** afterwards.
- Complete practice **RH124 CH 11.7.** (manual config)
- All settings done to NM or the ifcfg* files will be permanent, meaning they will be preserved across reboots. To temporarily set network settings (until reboot) or to work without NM → look at the next slide.

Manual configuration

- To manually set the ip address of an interface, issue a command like this: **ip addr add 192.168.50.5 dev eth1** To remove: **ip addr del 192.168.50.5 dev eth1**
- To enable/disable an interface, use **ip link set eth1 up/down**
- To add a routing table entry, issue **ip route add 10.10.20.0/24 via 192.168.50.100 dev eth0**
- To remove it, use: **ip route del 10.10.20.0/24 via 192.168.50.100 dev eth0**
- To set the default gateway: **ip route add default via 192.168.1.1**
- To manually edit the DNS resolver, modify /etc/resolv.conf with a text editor.
- Complete practice **RH124 CH 11.9.** (manual config)

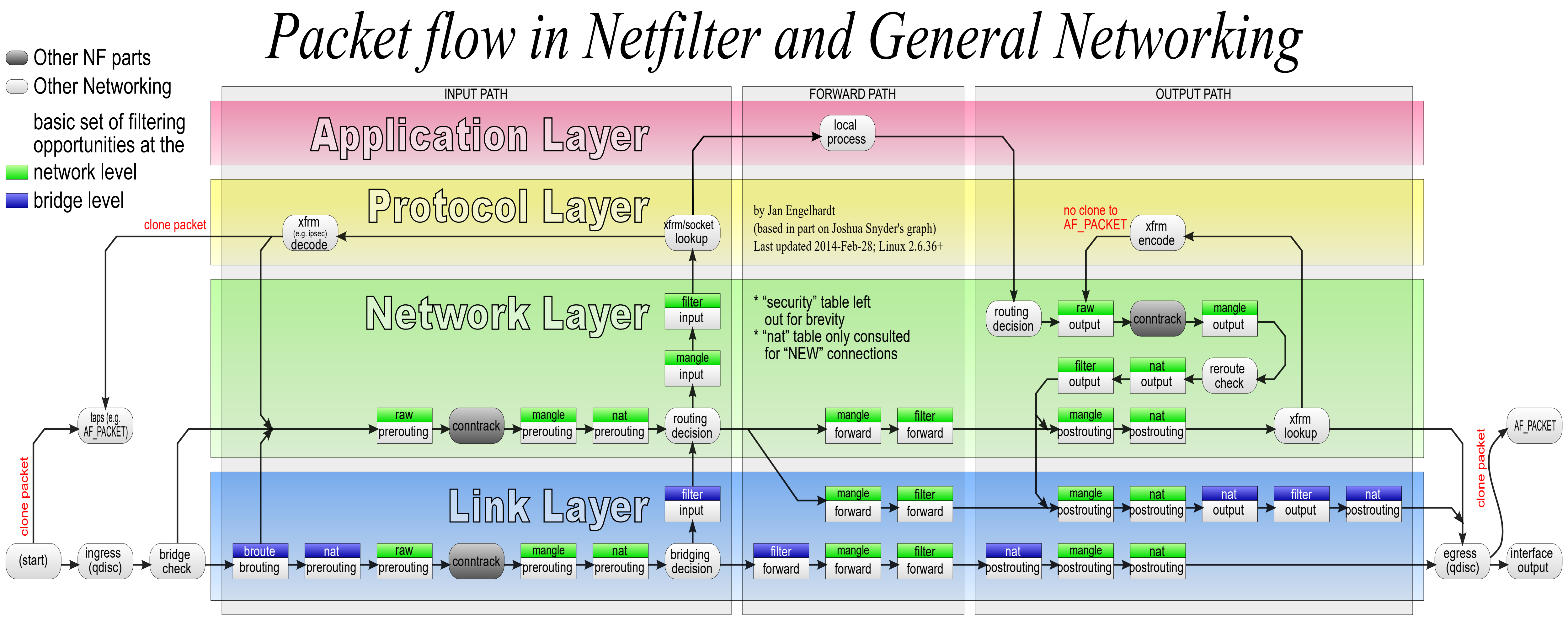
Firewall

Lecture 7

Chapter 2

Firewall functions

- In Linux the firewall functions are implemented in kernel space. This subsystem is called netfilter. The standard userspace tool for entering and manipulating the rule set is called iptables, ip6tables and ebtables.
- Firewall rules are applied to all network traffic that passes the kernel. All processes are subject to the ruleset, which is enforced by the kernel itself.
- In the RH family of Linux a user-friendly front-end called firewalld is responsible for loading the rule-set at boot time, and managing it later on. Firewalld resolves to calling several iptables commands behind the scenes.
- The older way of setting netfilter rules automatically was the iptables damon/script-set. You can either enable iptables or firewalld, but not both at the same time.
- Just like NM firewalld also has a GUI, TUI and CLI interface.



https://upload.wikimedia.org/wikipedia/commons/thumb/3/37/Netfilter-packet-flow.svg/800px-Netfilter-packet-flow.svg.png

Netfilter

- Netfilter is not only capable of filtering, but it can also manipulate packets and frames.
- Netfilter is a very complex and sophisticated system. Many network device manufacturers put an embedded Linux system in their devices, because it has a well built, proven, feature rich network stack.
- For e.g. the Linux kernel can implement a NAT/PAT router, which is achieved by setting the connection tracking and packet mangling rules accordingly.
- All the frames and packets come from network devices, which also work in kernel space. The whole netfilter job is done inside the kernel, therefore fast and reliable. Only the rule manipulation tools run in user space.
- Only a small part of the netfilter framework's features will be relevant to us.

iptables tables

- The rule-set of iptables consist of 5 tables: raw, filter, mangle, nat, security.
- Only two of those will be relevant for us: nat, where the PAT/NAT feature is implemented and filter, where the filtering rules apply.
- Mangle resembles nat, it can do specialized packet alterations.
- Security is only used by MAC (SELinux) mechanisms.
- Raw is a way of exempting sorts of traffic from connection tracking.

iptables chains

- Each table is built up of three major chains: the input, output, and forward chains. The rules in the chains are matched against traffic ACL style. The nat table also has a prerouting and a postrouting chain. Any number of chains can be added to tables manually.
- The INPUT chain controls the packets that are intended to be delivered to one of the addresses of the current system. (Not all received incoming packets are input)
- The FORWARD chain controls packets that are received as incoming data on one of the interfaces but are not meant for this specific system. If routing (IP forwarding) is enabled and configured correctly, such traffic will be forwarded via a relevant interface to another router/destination.
- The OUTPUT chain controls the packets that are assembled/originated by a process of the local system.

iptables rules

- Each chain contains a set of rules, that are evaluated in a specif order. A rule also specifies a target. Once a packet matches a rule, no further evaluation is done, it's target (fate) is decided.
- A target can be another chain, but most commonly it is a built-in target, like ACCEPT, DROP, REJECT, LOG, RETURN.
- Each chain has a default policy, which simply determines a target. If a packet does not match any rules, it will be destined to the default target.
- An example rule would be like the following:
  -A INPUT -p tcp --dport ssh -s 10.10.10.10 -j DROP
  This will destine all packets that come from 10.10.10.10 and have a destination port of 22 to the DROP fate.

iptables

- To add rules, simply issue **iptables *rule***, where rule is like a rule line shown above.
- Using the -D switch instead of -A will erase a rule.
- To clear all tables use **iptables -F**.
- To print all rules of a table: **ipatbles -L -t *tablename***, if omitting the -t switch, the FILTER table will be printed.
- The old fashioned iptables daemon simply saves all the active rules on shutdown and loads them at startup. This is done by calling /sbin/iptables-save and /sbin/iptables-restore. The I/O of these command should be redirected to a file, which holds the rules.
- In order for IP forwarding to work, besides the permissive rules in iptables you also have to enable the global packet forwarding switch in the kernel: **echo 1 > /proc/sys/net/ipv4/ip_forward**

iptables PAT router example

Shell command line

```
# iptables -F; iptables -t nat -F; iptables -t
mangle -F [this will clear the 3 relevant tables]
# iptables -t nat -A POSTROUTING -o ppp0 -j
MASQUERADE [this will turn on packet header
modification according to PAT. All TCP sessions
passing through will be tracked be the kernel.]
# iptables -A INPUT -m state --state ESTABLISHED,
RELATED -j ACCEPT
# iptables -A INPUT -m state --state NEW -i ! ppp0
-j ACCEPT
[only accept incoming packet that are related to an
established TCP session]
# iptables -P INPUT DROP
# iptables -A FORWARD -i ppp0 -o ppp0 -j REJECT
[new sessions initiated from outside will be
dropped, forwarding of foreign packages will be
rejected.]
# iptables -t nat -A PREROUTING -i ppp0 -p tcp
--dport 80 -j DNAT --to 172.31.0.23:80
[a port forward example]
```

Firewalld zones

- Firewalld divides traffic into zones. Zones are implemented as chains on the iptables level.
- Each zone has a separate list of rules (allowed services). An incoming packet will be classified to a zone, then matched against the zones rules and its fate will be determined.
- An incoming packet is checked for it's source address, to determine the applicable zone. If no zone rule matches to the source address, then the zone selection decision will be based upon the interface that received the packet. All interfaces belong to a zone. There is a default zone for new interfaces.
- Zones rank from trusted, work, external, ..., to dmz and drop, in order of strictness. You can even define your own zones.

firewalld services

- A service is defined by a set of ports, for e.g. the ssh service is equivalent to the single port of 22/tcp. The samba-client service consists of allowing two ports, 137/udp and 138/udp.
- A zone can have any number of enabled services. All ports, that do not belong to an enabled service of the zone, will not be available for communication.
- Use the **firewall-config** GUI utility for managing firewalld.
- Use **firewall-cmd** for command line management.
- You can print all the services with **firewall-cmd --get-services**.
- There is a way for enabling individual ports or ranges without assigning a service name to them.

firewall-cmd

- Firewalld separates it’s runtime configuration from the saved configuration that is automatically applied at startup. When modifying a setting it will only be applied to the runtime rule-set, but wont be saved to disk. Adding the **--permanent** switch to firewall-cmd will change the saved configuration, but not apply the rule at runtime.
- In many cases you have to issue a firewall-cmd command both with and without the --permanent switch.
- There is the **--timeout=N** switch for applying a change to firewall-cmd. A rule entered with the timeout switch will be automatically reverted after timeout passes. N is interpreted as number of seconds.
- Issuing **firewall-cmd --reload** will drop runtime configuration, reload and apply all settings of the saved state.

firewall-cmd

- Issue **firewall-cmd --get-zones** for listing the zones, **--get-active-zones** for printing only the zones that have at least one interface in them.
- Use the **--get-default-zone** switch to see the default zone for new interfaces.
- Use **--list-all-zones** to print detailed information about zones (for. e.g. interfaces, enabled services, etc.)
- Use **firewall-cmd --get-services** to list the available services.
- Issue **firewall-cmd --list-all** for printing the whole configuration.
- Use **firewall-cmd --set-default-zone=*zone*** to set the default zone to *zone.*

firewall-cmd

- Take the --**add-source=10.1.0.0/16 --zone=trusted** example to assign a network to a zone. Use **--remove-source=10.1.0.0/16 --zone=trusted** to revert such a change.
- Use the following example to add an interface to a zone: **firewall-cmd --add-interface=etho --zone=trusted** Use --change-interface=etho --zone=dmz, to reassign the interface to a different zone.
- Use **--add-service=mysql --zone=dmz** to enable a service such as mysql to be available in the dmz zone.
- Use **--add-port=17784/tcp --zone=dmz** to allow non-standard service ports to a zone, without creating a service for it.
- Use --remove-port and --remove-service logically.
- Complete practice **RH134 CH 14.2.** and **14.3.**

firewall-cmd

- To enable gateway functionality with firewalld:
  - Add your WAN facing interface to a zone, for e.g. external: **firewall-cmd --add-interface=pppo --zone=external --permanent**
  - Enable masquerading on the zone: **firewall-cmd --zone=external --add-masquerade --permanent**
  - Apply the configuration: **firewall-cmd --relaod**
- Take the following example: **firewall-cmd --add-forward-port=port=2822:proto=tcp:toport=22: toaddr=10.0.0.28 --zone=external** This exposes the ssh service of a masqueraded host: It will channel the data bound to the 2822 port of the gateway's external interface to the 22 port of the 10.0.0.28 machine behind the gateway. Instead of a single port you can also specify a port-range of the same size in *port* and *toport* fileds.
- Use --remove-forward-port=*[same long line]* to revert.

TCP Wrapper

Lecture 7

Chapter 3

TCP wrapper

- TCP wrapper is an implementation of ACLs for network services.
- It filters network access of processes, based on the peer's network address or hostname.
- TCP wrapper is a library, it works somewhat similar to libPAM. Binaries compiled against it can ask the "opinion" of the library, which in turn replies with a binary answer regarding whether the given communication can go on or not.
- As opposed to netfilter, which is enforced to all processes by the kernel, in case of the TCP wrapper library processes volunteer themselves for an ACL check.
- TCP wrapper is considerably easier to configure, than iptables, but almost as powerful in case you only want to limit access to services.

libwrap.so

- Most deamons from the repository are compiled to be libwrap-aware, libwrap.so is the name of the binary shared object file of TCP wrapper.
- You can check whether a process relies on libwrap.so or not by running ldd on it's binary executable file. This command will list all the dynamically linked libraries the given binary uses.

Shell command line

```
# which sshd
/usr/sbin/sshd
# ldd /usr/sbin/sshd
   linux-gate.so.1 =>  (0xb776c000)
→> libwrap.so.0 => /lib/libwrap.so.0 (0xb76a1000)
   libpam.so.0 => /lib/libpam.so.0 (0xb766b000)
   libssl3.so => /lib/libssl3.so (0xb6f92000)
   [...cut...]
   libz.so.1 => /lib/libz.so.1 (0xb73dc000)
```

TCP wrapper hosts files

- There are only two files considered by libwrap: the /etc/host.allow and /etc/hosts.deny text files.
- The .allow file is evaluated first, in case a match is not found, the .deny file is also parsed line by line. Once a match is found no further processing is done, the answer is decided (ACL style). In case no matches are found in either file, the access will be allowed.
- Even if the connection is allowed, it can be logged by libwrap.
- Each line in these files contains exactly one rule. The line format is the following:
  <daemon list> : <client list> [: <option> : <option> : ...]
- The last line should be terminated with a newline character.

hosts files format

- The daemon list is a comma separated list of process names (not service names) that the rule applies to.
- The list may also contain the ALL keyword and the EXCEPT operator. For e.g. ALL EXCEPT vsftpd
- The client list is a comma separated list of IP addresses, IP ranges or hostnames. Take the following examples:
  - .example.com -All hosts inside (ending with) the example.com domain name.
  - 192.168. -All IP addresses in the range of 192.168.0.0 – 192.168.255.255
  - 192.168.0.16/255.255.255.240 -An IP subnet
  - The ALL and EXCEPT keywords can also be used: .example.com EXCEPT attacker.example.com

hosts files format

- There are a few more wildcards to the client list, other than ALL.
- The LOCAL wildcard refers to hostnames without a period, like server7, localhost, etc…
- The PARANOID wildcard causes libwrap to do reverse DNS lookup based on the peers IP address. Then the resulting hostname is looked up as forward DNS query. If the result doesn't match, the access is denied. (spoofing)
- The KNOWN and UNKNOWN wildcard also relies on DNS, it is checked whether the host name and address is known or the user is known via identd.
- A number of options are possible, e.g. executing external commands on a match with the **spawn** option. The deny or allow option can also be used, this overrides the decision, that is normally determined by the filename.

TCP wrapper hosts examples

/etc/hosts.allow

```
sshd        : .example.com : spawn /bin/echo "$(/bin
/date) access denied" >> /var/log/sshd.log : deny

mysqld      : .company.net EXCEPT boss.company.net

in.ftpd     : 172.25.10.12,172.25.10.13

in.telnetd : /etc/telnet.hosts

ALL         : 192.168.
```

/etc/hosts.deny

```
sshd       : 10.10.9.9

ALL EXCEPT vsftpd : 10.16.32.0/255.255.224.0

mysqld     : [3ffe:505:2:1::]/64

ALL        : PARANOID
```

Recommended reading

Lecture 7

Chapter 4

Relevant chapters in RH:

- RH124 CH11 (Manage networking)
- RH134 CH14 (Firewalld)

Read by next lecture:

- RH124 CH10 (Logs)
- RH134 CH4 (Scheduling tasks)

Use your rhlearn.gilmore.ca login to access the learning materials.

Thank you for your attention!

Lecture 7, PM-TRTNB319, Linux System Administration

Zsolt Schäffer, PTE-MIK