

# Wireless network server



## Contents

Problem statement.....	3
Preliminary observation.....	4
Hardware.....	4
System installation and disk layout.....	7
Basic server configuration.....	8
Setting the system domain name.....	8
Disabling IPv6.....	8
Disabling Magic SysRq.....	9
Creation of the users.....	9
Configuring access control lists.....	11
Disabling unnecessary services.....	12
Configuring the list of repositories.....	14
Installing auxiliary software.....	14
Configuring network.....	15
Network interfaces configuration.....	15
Routing.....	17
Network bandwidth management.....	19
Wireless network setup.....	20
Managing a wireless network card.....	21
Configuring a software access point.....	24
Automatic changing of the access point password.....	26
Fail protection (auto-resume).....	27
Verification of performance and stability.....	27
DNS/DHCP service.....	29
Configuration.....	29
Fail protection (auto-resume).....	32
Verification of performance and stability.....	33
Network time service (ntp).....	34
Configuration.....	34
Fail protection (auto-resume).....	38
Verification of performance and stability.....	38
Installing and configuring ssh.....	39
Configuration.....	39

Fail protection (auto-resume).....	41
Verification of performance and stability.....	41
About additional services.....	42
Password protection (fail2ban).....	42
Log rotation (logrotate).....	43
Network time service (ntp).....	43
Mail service.....	43
Application software.....	45
Application software Suite and graphical environment.....	45
Automatic restoration of interface and program settings.....	46
Displaying the current password.....	47
Automatic system shutdown.....	47
Installing and configuring the network filter (nftables).....	47
Network environment and potential threats.....	47
The structure of the network filter.....	49
Starting and checking the network filter.....	50
Completing the installation.....	60
Using the server.....	60
Login procedure.....	60
Hardware health monitoring.....	61
Changing the external domain name server.....	62
Getting the current list of client devices.....	63
Notes on connecting clients.....	64

## Problem statement

It is needed to create a school wireless subnet server on the hardware base of the Lenovo g560 laptop according to the following network diagram:

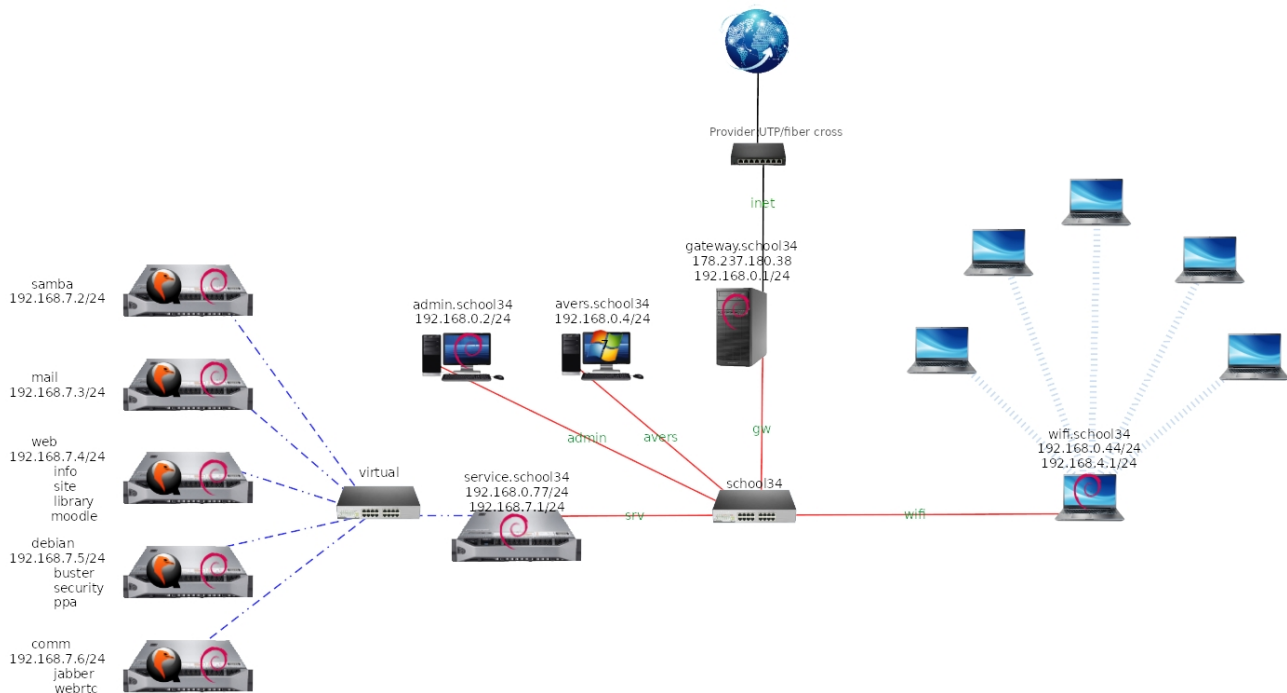


Figure 1: Network fragment

This diagram shows the part of the school's network that is visible to the wireless network server (server.wifi.school34), indicates domain names and ipv4 addresses of some nodes. Most of the network is hidden because it should not be visible to either the wireless network clients or the server itself. The figure also shows a virtualization server that serves the service.school34 domain. On the basis of this server a number of virtual machines are deployed to provide different network services to clients of the school's network. The core of the school's network includes the AVERS automated information system (AIS AVERS further in this document) server and the system administrator's computer.

The server must provide to clients a wireless transmission environment using an encrypted channel and regularly change the password to access it.

The wireless network server must provide DHCP service for client devices on its wireless interface, supporting up to 45 simultaneous clients. Addresses should be issued to them in the 192.168.4.0/24 subnet, taking into account that the 192.168.4.1 address is occupied by the server itself.

The server must provide domain name service to clients by forwarding requests for the service.school34 domain (192.168.7.0/24) to server.service.school34, for the school34 domain to gateway.school34, and all other requests to the external DNS server. In this case, it should be possible to quickly change the non-filtering (YandexDNS) external server to the filtering (SkyDNS) and back.

The server must provide precise time service to clients using gateway.school34, which is also a school-level precise time server and has the alias ntp.school34, for its own synchronization.

The server must be accessible for remote management only from the system administrator's computer (admin.school34).

The server must be able to act as a workstation, so it must be equipped with appropriate software: desktop environment (Plasma5), office software suite (LibreOffice, Okular, archivers and their interface, Internet browsers, raster and vector image editors, etc.).

Reasonable measures must be taken to protect the server and ensure the smooth operation of the server. Measures must also be taken to ensure the ergonomics of the graphical user environment.

## Preliminary observation

The following conventions are used in fragments of terminal sessions and when specifying commands to execute:

Commands that are executed when configuring the system are highlighted in bold, the system response or excerpts from configuration files and scripts are not highlighted in this way. All commands are provided with a full display of the command prompt, which reflects the user on whose behalf the command is executed, and the current working directory. Both of these factors are significant in most of the listed commands and listings.

At last, the original document was written in Russian, all commands and scripts were also executed in system with Russian localization. So it is not surprising that some messages obtained from system are localized too. There is no any translations for such messages in this document because of two reasons. Firstly, such localized fragments are not essential and can't preclude understanding the sense of the commands. Secondly, localized fragments serve as remark about the document origination.

## Hardware

The Lenovo g560 laptop from the company's existing machine fleet was chosen as the platform for implementing the wireless subnet server. The following abbreviated information is provided about the laptop hardware:

```
root@server:/# lshw
server
  description: Notebook
  product: 20042 (Calpella_CRB)
  vendor: LENOVO
  version: Lenovo G560
  serial: 2715387103105
  width: 64 bits
  capabilities: smbios-2.6 dmi-2.6 smp vsyscall32
...
*-core
  description: Motherboard
  product: Base Board Product Name
  vendor: LENOVO
  physical id: 0
  version: Base Board Version
  serial: CB09094437
```

```

    slot: Base Board Chassis Location
*-firmware
    description: BIOS
    vendor: LENOVO
    physical id: 0
    version: 29CN40WW(V2.17)
...
*-memory
    description: System Memory
    physical id: 19
    slot: System board or motherboard
    size: 4GiB
    *-bank:0
        description: SODIMM DDR3 Synchronous 1067 MHz (0,9 ns)
        product: RMT3020EF48E8W1333
        physical id: 0
        serial: 100EF037
        slot: DIMM0
        size: 2GiB
        width: 64 bits
        clock: 1067MHz (0.9ns)
    *-bank:1
        description: DIMM DDR3 Synchronous 1067 MHz (0,9 ns) [empty]
...
    *-bank:2
        description: SODIMM DDR3 Synchronous 1067 MHz (0,9 ns)
        product: HMT325S6BFR8C-H9
        physical id: 2
        serial: 2420BFD8
        slot: DIMM1
        size: 2GiB
        width: 64 bits
        clock: 1067MHz (0.9ns)
    *-bank:3
        description: DIMM DDR3 Synchronous 1067 MHz (0,9 ns) [empty]
...
*-cpu
    description: CPU
    product: Intel(R) Core(TM) i3 CPU           M 370  @ 2.40GHz
    vendor: Intel Corp.
    physical id: 2c
    bus info: cpu@0
    version: Intel(R) Core(TM) i3 CPU           M 370  @ 2.40GHz
    slot: CPU
    size: 1369MHz
    capacity: 2400MHz
    width: 64 bits
    clock: 1066MHz
    capabilities: lm fpu fpu_exception wp vme de pse tsc msr pae ...
    configuration: cores=2 enabledcores=2 threads=4
...
*-pci:0
    description: Host bridge
...
    *-display
        description: VGA compatible controller
        product: GT218M [GeForce 310M]
        vendor: NVIDIA Corporation
...
    *-multimedia
        description: Audio device
        product: High Definition Audio Controller

```

```

        vendor: NVIDIA Corporation
...
*-communication
    description: Communication controller
    product: 5 Series/3400 Series Chipset HECI Controller
...
*-multimedia
    description: Audio device
    product: 5 Series/3400 Series Chipset High Definition Audio
    vendor: Intel Corporation
...
*-pci:2
    description: PCI bridge
    product: 5 Series/3400 Series Chipset PCI Express Root Port 2
...
    *-network
        description: Wireless interface
        product: AR9285 Wireless Network Adapter (PCI-Express)
        vendor: Qualcomm Atheros
...
*-pci:3
    description: PCI bridge
    product: 5 Series/3400 Series Chipset PCI Express Root Port 3
...
    *-network
        description: Ethernet interface
        product: RTL8101/2/6E PCI Express Fast/Gigabit Ethernet controller
        vendor: Realtek Semiconductor Co., Ltd.
        physical id: 0
        bus info: pci@0000:07:00.0
        logical name: enp7s0
        version: 02
        serial: b8:70:f4:29:e5:83
        size: 100Mbit/s
        capacity: 100Mbit/s
...
*-sata
    description: SATA controller
    product: 5 Series/3400 Series Chipset 4 port SATA AHCI Controller
...
*-disk
    description: ATA Disk
    product: ST9320325AS
    physical id: 0
    bus info: scsi@0:0.0.0
    logical name: /dev/sda
    version: LVM1
    serial: 6VDCABKR
    size: 298GiB (320GB)
    capabilities: partitioned partitioned:dos
...

```

As comments to the information provided, please note the following:

The laptop has a central processor, the amount and characteristics of RAM, a hard disk, a video adapter, and other systems that are sufficient to perform the tasks.

Separately the device's network cards should be noted: the wired network interface only supports Fast Ethernet, and the lack of support for Gigabit Ethernet is not the best feature of the

device, but such network card is quite sufficient for given bandwidth restrictions imposed by the task statement.

A similar note can be made about a wireless network card: it is far from new, its signal level in the access point mode is sufficient to cover a very small area. However, this should not be considered as a disadvantage – even if it had a broader coverage, it would have to be limited due to the nature of the network, the nature of the organization itself, and the setting of the task.

For a detailed description of the characteristics of the wireless adapter, see the section "Organizing a wireless network" below.

It should also be noted that while closing the laptop cover (and the corresponding settings for the operating system and working environment were done), the wireless network card is not disabled.

## System installation and disk layout

The system is installed with the Plasma 5 graphical shell and a standard set of software for it, as well as with standard system utilities and ssh service.

The hard disk is divided into partitions as follows (the output below omits lines that do not concern the hard disk):

```
root@server:/# /usr/sbin/fdisk -l /dev/sda
Disk /dev/sda: 298,1 GiB, 320072933376 bytes, 625142448 sectors
Disk model: ST9320325AS
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x4bc7d946
```

Device	Boot	Start	End	Sectors	Size	Id	Type
/dev/sda1	*	2048	58593279	58591232	28G	83	Linux
/dev/sda2		58593280	117186559	58593280	28G	83	Linux
/dev/sda3		117186560	124999679	7813120	3,7G	82	Linux swap / Solaris
/dev/sda4		124999680	625141759	500142080	238,5G	83	Linux

```
root@server:/# /sbin/parted -l /dev/sda
Model: ATA ST9320325AS (scsi)
Disk /dev/sda: 320GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:
```

Number	Start	End	Size	Type	File system	Flags
1	1049kB	30,0GB	30,0GB	primary	ext4	boot
2	30,0GB	60,0GB	30,0GB	primary	ext4	
3	60,0GB	64,0GB	4000MB	primary	linux-swaps(v1)	
4	64,0GB	320GB	256GB	primary	ext4	

```
root@server:/# mount -l
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-
ro,jqfmt=vfsv0,usrjquota=aquota.user) [rootfs]
/dev/sda2 on /var type ext4
(rw,nosuid,nodev,relatime,jqfmt=vfsv0,usrjquota=aquota.user) [var]
/dev/sda4 on /home type ext4
(rw,nosuid,nodev,relatime,jqfmt=vfsv0,usrjquota=aquota.user) [home]
```

```
root@server:/# df -h
```

Файловая система	Размер	Использовано	Дост	Использовано%	Смонтировано в
/dev/sda1	28G	8,0G	19G	31%	/
/dev/sda2	28G	991M	25G	4%	/var
/dev/sda4	234G	275M	222G	1%	/home

This disk layout scheme was chosen for the following reasons:

The DOS table is the simplest and sufficient for the server, especially considering the hardware release year. The 4 GB swap partition is allocated, and the rest of the disk space is divided into three partitions, so that the /var directory and users' home directories are located in separate file systems. This is done as part of basic server protection measures. Most of the disk space is reserved for users' home directories, while the rest of the system components are allocated with minimal disk space (with some margin).

## Basic server configuration

This section describes the measures that apply to many of servers like this one. In particular, we consider ways to minimize the number of services, configure the network subsystem and some security systems.

### Setting the system domain name

Since the system is not supposed to use the IPv6 protocol, to configure the full domain name, first fill in the **/etc/hosts** file, then enter the system name without the domain part in the **/etc/hostname** file, and run the command to change the host name in the current session. All the necessary commands, as well as the contents of the configuration files, are presented in the following fragment of the terminal session. The last command in it shows the configured full domain name of the system (FQDN):

```
root@server:/# cat /etc/hosts
127.0.0.1      localhost.localdomain localhost
127.0.1.1      server.wifi.school34 server
root@server:/# cat /etc/hostname
server
root@server:/# hostname -f
server.wifi.school34
```

### Disabling IPv6

Disabling IPv6 support by the system core can be performed either for the current session or on a permanent basis. To permanently disable it, make the contents of the **/etc/sysctl.conf** file look like so that it don't contradict the following lines:

```
#Disabling Ipv6
net.ipv6.conf.all.disable_ipv6=1
net.ipv6.conf.default.disable_ipv6=1
net.ipv6.conf.lo.disable_ipv6=1
```

To disable it only in the current session run the following commands:

```
root@server:/# echo '1' > /proc/sys/net/ipv6/conf/all/disable_ipv6
root@server:/# echo '1' > /proc/sys/net/ipv6/conf/default/disable_ipv6
root@server:/# echo '1' > /proc/sys/net/ipv6/conf/lo/disable_ipv6
```



To check whether the IPv6 shutdown is successful, restart the network subsystem with the command

```
root@server:/# service networking restart
```

and in the output of the command

```
root@server:/# ip address show
```

ensure that no IPv6 addresses are assigned:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    link/ether b8:70:f4:29:e5:83 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.44/24 brd 192.168.0.255 scope global enp7s0
        valid_lft forever preferred_lft forever
3: wlp6s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default qlen 1000
    link/ether d0:df:9a:6a:89:18 brd ff:ff:ff:ff:ff:ff
    inet 192.168.4.1/24 brd 192.168.4.255 scope global wlp6s0
        valid_lft forever preferred_lft forever
```

Note: this is the output when the network subsystem is fully configured. A description of these settings is given below, here it suffices to note only the lack of interfaces' ipv6 addresses.

## Disabling Magic SysRq

Disabling Magic SysRq may seem redundant or even harmful: for example, if the system crashes, it will not be possible to make emergency shutdown or dump the contents of buffers to disk. However, keep in mind that the server is not intended for storing important data. At the same time, due to the physical availability of the server to users, it is likely that some attacker will perform a similar key combination. This can lead to server shutdown and client service failures. At the same time, protection is organized so simply that it is not rational to neglect it.

To disable "magic" combinations in the current session, just use the command:

```
root@debian:/# echo '0' > /proc/sys/kernel/sysrq
```

To disable it permanently, you just need to make the `/etc/sysctl.conf` file look like it doesn't contradict the next line

```
kernel.sysrq=0
```

## Creation of the users

Further, when describing the secure remote SSH login service, it is indicated that logging in remotely as a superuser is blocked. To still be able to log in remotely, it is needed to create a regular user, on whose behalf the remote login will be performed. A session of the graphical user interface will also be loaded for this user, which allows you to use the server as a workstation. After opening a session in the system under such account, it is possible to increase, if necessary,

credentials using su. There are no limits on system resources for this user at the moment, but based on the experience of using the system, this can be done in the future.

It is possible to create such user (let's say its username is guest) both at the system installation stage and after, for example, using the command

```
root@debian:/# /usr/sbin/adduser guest
```

In addition to creating a user, it is needed to pay attention to several aspects of his operation in the system.

First, by default, the newly created home directory of this user is readable by everyone in the system. If this is not acceptable, you can correct the situation using the command

```
root@debian:/# chmod 0750 /home/guest/
```

thus leaving full access to the directory for the user and read access for the members of his group (who are not present in the system except him). Access will be closed for anybody else.

The second aspect is the use of sudo. This command allows you to execute commands with administrative privileges without logging in as a superuser. This approach is fraught with some danger: an attacker knowing the username and password of such user can get very extensive rights in the system (if sudo policies are configured) or almost unlimited if sudo settings are accepted by default. To solve this problem, it is the easiest way (and most optimal on a similar server with a single user and a minimum number of services) to use the following approach: do not use sudo at all, forcing the user to increase his rights in the system by entering (and, accordingly, knowing) the superuser password.

Thus, to prevent the user from using sudo, it is enough to exclude him from the wheel group, if he is a members of it. You can do this with the usermod command, leaving the guest user only in his (namesake) group. To make sure that the changes are successful, run the groups command on behalf of this user strictly after logging in again. The following is a fragment of the terminal session that demonstrates all these operations:

```
root@server:/# /usr/sbin/usermod -G guest guest
root@server:/# su guest
guest@server:/$ groups
guest
guest@server:/$ exit
exit
root@server:/#
```

You should also set disk quotas for the user on file systems whose overflow may be sensitive to system health. In the system under consideration, quotas are configured as follows:

```
root@server:/# setquota -u guest 10G 10G 0 0 /
root@server:/# setquota -u guest 240G 240G 0 0 /home
```

With these quotas, the guest user is allowed to use not more than 10 GB of the 30 available in the root file system (where the /tmp and /var/tmp directories are located) and not more than 240 GB of the 256 available in the home directory section.

You can make sure that quotas are set in the output of the command

```
root@server:/# quota -s --show-mntpoint guest
Disk quotas for user guest (uid 1001):
    Filesystem    space   quota   limit   grace   files   quota   limit   grace
```

/dev/sda1 /	600K	10240M	10240M	151	0	0
/dev/sda4 /home	242M	240G	240G	4254	0	0

The final stage of creating a guest user should be considered to ensure that it is possible to log in to the graphical environment of the system's desktop without entering password, which is implied by the nature of the system's use. The steps required for this are described later in the section "Application software".

## Configuring access control lists

Many daemons use the **/etc/hosts.allow** and **/etc/hosts.deny** files as sources of information about who is allowed to use the services of these daemons. These configuration files are part of the tcp wrappers mechanism. In addition, a tcpd daemon can be installed on the system, which performs this check itself before passing the network packet that initiates the connection to the target daemon.

To find out whether the daemon of a certain service uses this mechanism, it is usually enough to make sure that the daemon executable is built using the libwrap library. The following fragment of the terminal session demonstrates that the sshd daemon uses this library, whereas dnsmasq and ntpd do not:

```
root@server:/# ldd /sbin/ntpd | grep libwrap
root@server:/# ldd /sbin/dnsmasq | grep libwrap
root@server:/# ldd /sbin/sshd | grep libwrap
libwrap.so.0 => /lib/x86_64-linux-gnu/libwrap.so.0 (0x00007fa52ea65000)
```

To check whether the tcpd daemon is installed on the system, use the command

```
root@server:/# dpkg -l | grep tcpd
```

As you can see from the empty command response, this daemon is not installed.

In general, this check is performed by the firewall, and the tcp wrapper mechanism is considered outdated and has given way to network screens. However, keeping in the mind the ease of configuration and low resource requirements, it makes sense to make entries in the specified configuration files for all the demons used by the system, thus leaving the decision to use this mechanism to the daemon developers and distribution assemblers. In other words, if they decide to enable support in a new version of the software, the customized system will automatically be ready for such changes. In addition, no software is absolutely immune from developer errors and vulnerabilities. The use of this mechanism may be an additional line of defense in case of the network filter compromise.

In this system it is senseless to install tcpd, since ssh uses this mechanism independently, and other daemons simply do not accept connections on the external (wired) interface.

Knowing that ssh connection is allowed only from the system administrator's computer with the address 192.168.0.2, and ntp and dns/dhcp only from the internal (wireless) interface, you should bring the **/etc/hosts.allow** file to the following content (comments are omitted for brevity):

```
ntpd : 192.168.4.0/24 127.0.0.1
dnsmasq : 192.168.4.0/24 127.0.0.1
sshd : 192.168.0.2
```

In turn, the **/etc/hosts.deny** file prohibits any other connections to the server (comments are also not provided):

ALL: ALL

After restarting the services, you can make sure that ssh does not accept connections from foreign hosts by restarting the service with the command

```
root@debian:/# systemctl restart sshd
```

and trying to connect via ssh from any host other than the subnet server. The firewall should not block this connection at the time of the experiment.

Of course, you should configure such security parameters before connecting the server to the network, and finally check them immediately after, if possible, protecting the network segment being checked by other means (for example, using the firewall of the organization's higher-level server).

## Disabling unnecessary services

Following the necessary sufficiency rule, you should disable and/or delete all unused services, for example: rsh, telnet, rpcbind. To check whether these services are installed, use one of the following commands or a combination of the following (and possibly some additional) commands:

```
root@server:/# dpkg -l | grep rpcbind
root@server:/# which rsh
root@server:/# ss -l | grep telnet
```

None of these methods is universal in the sense that it clearly shows the presence or absence of a particular service. The following fragment of a terminal session can be used as an example:

```
root@server:/# which rsh
/usr/bin/rsh
root@server:/# ss -Htpl
LISTEN 0 32 127.0.0.1:domain 0.0.0.0:* users:(("dnsmasq",pid=745,fd=9))
LISTEN 0 32 192.168.4.1:domain 0.0.0.0:* users:(("dnsmasq",pid=745,fd=7))
LISTEN 0 128 192.168.0.44:ssh 0.0.0.0:* users:(("sshd",pid=738,fd=3))
LISTEN 0 5 127.0.0.1:ipp 0.0.0.0:* users:(("cupsd",pid=613,fd=7))
LISTEN 0 50 *:1716 *: users:(("kdeconnectd",pid=946,fd=12))
root@server:/# dpkg -l | grep rsh
ii hershey-font-gnuplot 0.1-1+b1 amd64 Hershey vector fonts renderer for gnuplot
ii hershey-fonts-data 0.1-1 all Hershey vector fonts collection
ii libhersheyfont0 0.1-1+b1 amd64 Hershey vector fonts shared library
root@server:/# ls -l /usr/bin | grep rsh
-rwxr-xr-x 1 root root 10512 dek 3 2013 hershey-font-gnuplot
lrwxrwxrwx 1 root root 21 ceH 17 20:38 rsh -> /etc/alternatives/rsh
root@server:/# ls -l /etc/alternatives | grep rsh
lrwxrwxrwx 1 root root 12 ceH 17 20:38 rsh -> /usr/bin/ssh
lrwxrwxrwx 1 root root 28 ceH 17 20:38 rsh.1.gz -> /usr/share/man/man1/ssh.1.gz
root@server:/#
```

Analyzing the given fragment, you can note the following:

The first command indicates that the rsh executable file is present on the system, but the second command indicates that the rsh daemon does not listen for any tcp ports. The following command informs you that there are 3 packages installed in the system that mention rsh in their names, but it is obvious from their names that these packages are not related to rsh. When viewing the properties of an executable file in detailed mode, it turns out that it is a symbolic link

to another file, which is also a symbolic link to the ssh executable. Thus, rsh is not installed on this system, instead an alias for ssh is created.

Among other things, rpcbind and nfs-common packages were installed in the system as part of the graphical working environment. since deploying an nfs server on the machine is not part of the task statement, as well as connecting this machine to other nfs servers, both of these packages can be deleted by the command:

```
root@server:/# apt-get purge rpcbind
```

This frees up the system resources that they occupy and closes unused ports.

You should also delete the desktop service for connecting mobile devices (KDE Connect) and the Avahi system for detecting services on the local network. Avahi is useless on this machine because it is the wireless subnet server that provides all the necessary services, and from the point of view of the network core, the wireless subnet server is a client, but all the services it needs are notified in advance.

You can delete unnecessary services with commands (insignificant output of commands is reduced):

```
root@server:/# apt-get purge kdeconnect
```

```
Чтение списков пакетов... Готово
```

```
...
```

```
Следующий пакет устанавливался автоматически и больше не требуется:
```

```
libfakekey0
```

```
Для его удаления используйте «apt autoremove».
```

```
root@server:/# apt-get purge avahi-daemon
```

```
root@server:/# apt-get purge avahi-autoipd
```

```
root@server:/# apt autoremove
```

After restarting the desktop session, you can make sure that the system now listens only to the ports it needs to perform its tasks:

```
root@server:/# ss -tulp
```

Netid	State	Recv-Q	Send-Q	Local Address:Port	users:
udp	UNCONN	0	0	0.0.0.0:40203	users:(("dnsmasq",pid=691,fd=10))
udp	UNCONN	0	0	192.168.4.1:domain	users:(("dnsmasq",pid=691,fd=6))
udp	UNCONN	0	0	127.0.0.1:domain	users:(("dnsmasq",pid=691,fd=8))
udp	UNCONN	0	0	0.0.0.0:bootps	users:(("dnsmasq",pid=691,fd=4))
udp	UNCONN	0	0	192.168.0.44:ntp	users:(("ntpd",pid=666,fd=18))
udp	UNCONN	0	0	192.168.4.1:ntp	users:(("ntpd",pid=666,fd=22))
udp	UNCONN	0	0	127.0.0.1:ntp	users:(("ntpd",pid=666,fd=17))
udp	UNCONN	0	0	0.0.0.0:ntp	users:(("ntpd",pid=666,fd=16))
udp	UNCONN	0	0	0.0.0.0:ipp	users:(("cups-browsed",pid=489,fd=7))
tcp	LISTEN	0	32	192.168.4.1:domain	users:(("dnsmasq",pid=691,fd=7))
tcp	LISTEN	0	32	127.0.0.1:domain	users:(("dnsmasq",pid=691,fd=9))
tcp	LISTEN	0	128	192.168.0.44:ssh	users:(("sshd",pid=670,fd=3))
tcp	LISTEN	0	5	127.0.0.1:ipp	users:(("cupsd",pid=479,fd=6))

Note the UDP port 40203, which is opened by dnsmasq itself for accessing higher-level dns servers. The number of this port changes when the system is restarted, and dnsmasq itself does not accept client requests on it.

Also of note is the CUPS printer management service and its cups-browsed and cupsd daemons. At the time of writing this document, it is not intended to connect printers to the system in question, or even provide printing services to customers. However, such an extension of

functionality is very likely, so the service is not blocked, especially since with the default settings, the printing service (cupsd) listens only to the local loop interface. The cups-browsed daemon, in turn, only listens to the network while waiting for a broadcast from print servers that are ready to provide their services. In such circumstances you only need to close the print service port using a firewall as will be shown later.

## Configuring the list of repositories

Immediately after installing the system to be able to install additional software, fill in the list of repositories used by the system, register public parts of local repository keys, and configure the system's wired network adapter. The description of configuring network interfaces is given below to preserve the consistency and integrity of the presentation. There are no obstacles to configuring the repository list and installing additional software after configuring the wired network.

The `/etc/apt/sources.list` configuration file that contains a list of repositories should look like:

```
deb http://debian.service.school34/buster/ buster main contrib non-free
deb http://debian.service.school34/security/ buster/updates main contrib
deb http://debian.service.school34/ppa/ buster main contrib
```

Then you need to get the public part of the local repository key (ppa), for example, by downloading the `repository_key.asc` file from it with the command:

```
root@server:/# cd /tmp && wget http://debian.service.school34/ppa/repository_key.asc
```

Next, this file should be registered in the package management system with the command:

```
root@server:/tmp# apt-key add repository_key.asc
```

After that, you can update information about packages stored in the specified repositories:

```
root@server:/tmp# apt-get update
```

You don't need to import keys from other repositories, because they are mirrors of official repositories and are installed with the system.

## Installing auxiliary software

The Midnight Commander file manager (mc package) and the tree utility can be very useful when maintaining the server. To install them just run the command:

```
root@server:/# apt-get install mc tree
```

Viewing data from hardware sensors is very useful for monitoring the server state. These tools are part of the lm-sensors package, which you need to install and then perform the process of determining the sensors available in the system. You can do this by running the following commands and following the instructions that appear:

```
root@server:/# apt-get install lm-sensors
root@server:/# sensors-detect
```

Then you can get the data from these sensors using the command

```
root@server:/# sensors
```

Moreover, this command can be executed even with the rights of an unprivileged user.

Another means of monitoring the system state is the S. M. A. R. T technology – assessment of the state of the hard disk (or other storage) with built-in self-diagnostics equipment.

To use this technology, you must install the smartmontools package, which contains utilities that allow you to retrieve data from the hard disk, interpret and display it, and perform disk testing. This package also includes the smartd daemon (and the corresponding systemd service), which automatically monitors the disk status and informs the system administrator. For more information, see the section "Using the server".

In addition, it is very convenient to use the GSmartControl program – an utility for working with SMART, which has a graphical user interface.

You can install these software tools using the following commands:

```
root@server:/# apt-get install smartmontools
root@server:/# apt-get install gsmartcontrol
```

Their application is shown in the section "Using the server".

## Configuring network

This section only describes configuring the network interfaces of the system itself. the description of creating a wireless access point is given below. Network configuration can be divided into the following steps:

- installing the necessary software;
- configuring network interfaces;
- configuring packet routing and forwarding;
- configuring the bandwidth management system;

For the full operation of the wireless network card in the system under consideration, you need to install the firmware-atheros package. This can be done immediately after configuring the wired network interface, or by manually transferring the package from a removable media. If the network interface and the list of repositories are configured, you can perform the installation using the command:

```
root@server:/# apt-get install firmware-atheros
```

The rest of the network configuration tools (iproute2, tc) are installed with the system.

## Network interfaces configuration

The system has two network interfaces: wired enp7s0 and wireless wlp6s0. This can be seen from the output of the following command:

```
root@server:/# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
   group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
   DEFAULT group default qlen 1000
    link/ether b8:70:f4:29:e5:83 brd ff:ff:ff:ff:ff:ff
3: wlp6s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode
```

```
DEFAULT group default qlen 1000
    link/ether d0:df:9a:6a:89:18 brd ff:ff:ff:ff:ff:ff
```

You can configure network interfaces by entering the necessary information in the **/etc/network/interfaces** file. In this case it looks like (with some insignificant abbreviations):

```
source /etc/network.interfaces.d/*
```

```
#The loopback network interface
auto lo
iface lo inet loopback
```

```
#External network interface
auto enp7s0
iface enp7s0 inet static
    address 192.168.0.44/24
    gateway 192.168.0.1
    pre-up /etc/network/neighbors
    pre-up /etc/network/shaping
    up      /etc/network/routes
```

```
#Internal network interface
auto wlp6s0
iface wlp6s0 inet static
    address 192.168.4.1/24
```

Thus, wired network interface, acting as an external (relative to customers) gets the address in the core network and wireless (internal) address in the wireless subnet.

To apply the new configuration restart the network subsystem with the command:

```
root@server:/# systemctl restart networking
```

Then make sure that the restart is successful via the command output:

```
root@server:/# systemctl status networking
• networking.service - Raise network interfaces
  Loaded: loaded (/lib/systemd/system/networking.service; enabled; vendor preset:
enabled)
  Active: active (exited) since Mon 2020-03-16 13:51:51 MSK; 40s ago
  Docs: man:interfaces(5)
  Process: 1499 ExecStart=/sbin/ifup -a --read-environment (code=exited,
status=0/SUCCESS)
  Main PID: 1499 (code=exited, status=0/SUCCESS)

map 16 13:51:45 server systemd[1]: Starting Raise network interfaces...
map 16 13:51:51 server systemd[1]: Started Raise network interfaces.
```

Finally, you can see the new network settings of the system in the output of the command:

```
root@server:/# ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc htb state UP group
default qlen 1000
    link/ether b8:70:f4:29:e5:83 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.44/24 brd 192.168.0.255 scope global enp7s0
        valid_lft forever preferred_lft forever
```



```

3: wlp6s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default qlen 1000
    link/ether d0:df:9a:6a:89:18 brd ff:ff:ff:ff:ff:ff
    inet 192.168.4.1/24 brd 192.168.4.255 scope global wlp6s0
    valid_lft forever preferred_lft forever

```

As you can see from this output, there are three network interfaces in the system, counting the loop one, they use exactly the addresses that were assigned to them, and IPv6 addresses are not assigned. Thus, the network interface settings fully meet the requirements set above.

In addition, three scenarios are performed before and during the launch of the external network interface: entering static entries in link-layer address tables, configuring routing, and configuring bandwidth for client connections.

All scripts are located in the same **/etc/network** directory as the main network interface configuration file, and are readable, writable, and executable only by the superuser. Creating these files and setting their attributes can be performed by commands (the output of the last command shows only the files described):

```

root@server:/# cd /etc/network
root@server:/etc/network# touch shaping routes neighbours
root@server:/etc/network# chmod 0700 shaping routes neighbours
root@server:/etc/network# ls -l
...
-rw-r--r-- 1 root root  548 map 25 10:48 interfaces
...
-rwx----- 1 root root  331 map 24 13:49 neighbours
-rwx----- 1 root root   67 map 25 10:51 routes
-rwx----- 1 root root 1197 feb 26 14:18 shaping

```

At the moment the script for creating static arp records looks like this:

```

#!/bin/sh

ip neighbour add 192.168.0.1 dev enp7s0 lladdr 00:c0:26:a7:b9:39 nud permanent
ip neighbour add 192.168.0.2 dev enp7s0 lladdr 90:2b:34:48:08:b5 nud permanent
ip neighbour add 192.168.0.4 dev enp7s0 lladdr 94:de:80:bc:33:de nud permanent
ip neighbour add 192.168.0.77 dev enp7s0 lladdr b8:70:f4:22:b0:6b nud permanent

```

As hardware is replaced on the virtualization server and main gateway, their hardware addresses must be updated.

The output of the following command, executed after restarting the network subsystem, demonstrates checking static records for hardware addresses:

```

root@server:/# ip neighbour show
192.168.0.4 dev enp7s0 lladdr 94:de:80:bc:33:de PERMANENT
192.168.0.2 dev enp7s0 lladdr 90:2b:34:48:08:b5 PERMANENT
192.168.0.1 dev enp7s0 lladdr 00:c0:26:a7:b9:39 PERMANENT
192.168.0.77 dev enp7s0 lladdr b8:70:f4:22:b0:6b PERMANENT

```

## Routing

It should be noted that device communication in the organization's local network is implemented by splitting the local network into isolated subnets protected by their own firewalls, which simultaneously act as servers for such subnets. One of these servers (for a wireless subnet) is the system in question. These servers are responsible for routing packets and performing address

translation (NAT). From the point of view of routing, each such server should, on the one hand, provide clients with the shortest route to the target node of the network, and on the other – not provide a route to those subnets that are not provided for by the logic of the network and the organization. In this case, the wireless network server should only provide clients with routes to the network core (including the main gateway and the AIS AVERS server) and to the virtual servers' network. You must also provide a default route through the organization's main gateway. This section does not address issues of protection against attempts to get into other subnets using the routing tables of other servers, for example, the main gateway, because protection against such attacks is the responsibility of the firewall of this gateway itself. Next, the section "Installing and configuring the network filter (nftables)" describes measures that prevent using this server for the same purpose.

Regarding the issue of address translation, it should be noted that NAT has no effect on routing within the local network. When accessing external resources, symmetric NAT is performed on the main gateway of the organization. Within a local network, such transformation can have both positive and negative features in its application. Negative aspects include the increased load on the machine performing this conversion, and the inability to find out the client's source ip address in server logs. This same impossibility in some cases should be considered a virtue. For example, by using NAT on the accounting subnet server, you can achieve a fairly reliable hiding of information about the number of hosts in this subnet, their internal addresses, the ports used by each of them, and so on. As a result, it will be impossible to determine which accounting computer was connected to the local information server. If the same mechanism is applied to a wireless subnet, the result will be hiding traces of the malicious client in the logs of the same info server. This is why NAT is not used in a wireless network.

First of all, when configuring routing, you should ensure that network packets are forwarded between the interfaces of the server in question. To do this, make the configuration file **/etc/sysctl.conf** to contain the string

```
net.ipv4.ip_forward=1
```

Usually, it is enough to simply uncomment it. To enable packet forwarding in the current session without restarting the system just run the command:

```
root@server:/# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Then add the necessary network routes in order to send packets along the shortest path. To do this, in the previously mentioned routing configuration scenario, you just need to bring the following:

```
root@server:/# cat /etc/network/routes
#!/bin/sh
```

```
ip route add 192.168.7.0/24 via 192.168.0.77 dev enp7s0
```

After restarting the network subsystem, you can use the following commands to make sure that the added route is both available and working:

```
root@server:/# ip route show
default via 192.168.0.1 dev enp7s0 onlink
192.168.0.0/24 dev enp7s0 proto kernel scope link src 192.168.0.44
192.168.4.0/24 dev wlp6s0 proto kernel scope link src 192.168.4.1
192.168.7.0/24 via 192.168.0.77 dev enp7s0
root@server:/# traceroute 192.168.7.1
```

```

traceroute to 192.168.7.1 (192.168.7.1), 30 hops max, 60 byte packets
 1  server.service.school34 (192.168.7.1)  0.524 ms  0.411 ms  0.316 ms
root@server:~# ping -c 3 debian.service.school34
PING debian.service.school34 (192.168.7.5) 56(84) bytes of data:
64 bytes from debian.service.school34 (192.168.7.5): icmp_seq=1 ttl=63 time=2.22 ms
64 bytes from debian.service.school34 (192.168.7.5): icmp_seq=2 ttl=63 time=1.92 ms
64 bytes from debian.service.school34 (192.168.7.5): icmp_seq=3 ttl=63 time=2.04 ms

--- debian.service.school34 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 5ms
rtt min/avg/max/mdev = 1.916/2.061/2.224/0.126 ms

```

The routing table provided by the first command also contains routes created based on network interface settings. Together, they make up the complete set of routes that the server needs to work with.

## Network bandwidth management

Bandwidth settings for client connections are performed using the traffic control mechanism and the tc utility based on the following considerations:

- traffic is divided into three classes: connections to the AVERS server, connections to the internal virtualization server, and connections to Internet resources
- for traffic to the AVERS server 20 Mbit/s bandwidth is guaranteed with the possibility of extending it to 50 Mbit/s
- for traffic to external Internet resources 1 Mbit/s bandwidth is guaranteed with the possibility of extending it to 10 Mbit / s
- 40 Mbit/s bandwidth is guaranteed for traffic to internal servers and can be extended to 100 Mbit/s
- bandwidth for traffic to the AVERS server is limited to 50 Mbit/s in order to prevent DoS/DDoS attacks on this server by wireless network clients
- bandwidth to external Internet resources is limited on top by the width of the external channel.
- bandwidth for traffic to the internal servers of the school is limited from above by the bandwidth of the wired network interface of the described system
- overcommitting, i.e. setting the total guaranteed bandwidth that exceeds the hardware capabilities of the interface, is not allowed

Based on these considerations, the scenario for configuring bandwidth for transit network connections takes the form:

```

#!/bin/sh

#Resetting current state
tc qdisc delete dev enp7s0 root
#Setting root discipline
tc qdisc add dev enp7s0 root handle 1: htb default 13
#Creating root class
tc class add dev enp7s0 parent 1: classid 1:1 htb rate 100mbit ceil 100mbit
#Creating subclasses: avers, service, inet
tc class add dev enp7s0 parent 1:1 classid 1:11 htb rate 20mbit ceil 50mbit
tc class add dev enp7s0 parent 1:1 classid 1:12 htb rate 40mbit ceil 100mbit
tc class add dev enp7s0 parent 1:1 classid 1:13 htb rate 1mbit ceil 10mbit
#Configuring disciplines for subclasses
tc qdisc add dev enp7s0 parent 1:11 handle 10:0 sfq perturb 10
tc qdisc add dev enp7s0 parent 1:12 handle 20:0 sfq perturb 10

```

```
tc qdisc add dev enp7s0 parent 1:13 handle 30:0 sfq perturb 10
#Configuring filters for traffic classification (inet by default)
tc filter add dev enp7s0 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0.4
flowid 1:11
tc filter add dev enp7s0 protocol ip parent 1:0 prio 1 u32 match ip dst
192.168.7.0/24 flowid 1:12
```

First of all, the script deletes the current root discipline along with all its components and connects the htb (Hierarchical Token Bucket) discipline as the root. It specifies that all unclassified (default) traffic must be processed using the 1:13 (Internet traffic) disciplines. Then the root class is created, where all traffic will go (this is necessary for implementing borrowing). In this class, bandwidth is limited according to hardware capabilities. Next, three subclasses are created that provide the specified channel width separation. After that, three sfq (Stochastic Fairness Queueing) disciplines are created, one for each class. Finally, two filters are created: the first classifies traffic to the AIS AVERS server, and the second classifies traffic to the virtualization server. Unclassified traffic is considered to be directed to the Internet.

After running the script, you can check the status of the traffic control subsystem with the following commands:

```
root@server:/# tc qdisc show
qdisc noqueue 0: dev lo root refcnt 2
qdisc htb 1: dev enp7s0 root refcnt 2 r2q 10 default 0x13 direct_packets_stat 0
direct_qlen 1000
qdisc sfq 30: dev enp7s0 parent 1:13 limit 127p quantum 1514b depth 127 divisor 1024
perturb 10sec
qdisc sfq 10: dev enp7s0 parent 1:11 limit 127p quantum 1514b depth 127 divisor 1024
perturb 10sec
qdisc sfq 20: dev enp7s0 parent 1:12 limit 127p quantum 1514b depth 127 divisor 1024
perturb 10sec
qdisc noqueue 0: dev wlp6s0 root refcnt 2
root@server:/# tc class show dev enp7s0
class htb 1:11 parent 1:1 leaf 10: prio 0 rate 20Mbit ceil 50Mbit burst 1600b cburst
1600b
class htb 1:1 root rate 100Mbit ceil 100Mbit burst 1600b cburst 1600b
class htb 1:13 parent 1:1 leaf 30: prio 0 rate 1Mbit ceil 10Mbit burst 1600b cburst
1600b
class htb 1:12 parent 1:1 leaf 20: prio 0 rate 40Mbit ceil 100Mbit burst 1600b cburst
1600b
root@server:/# tc filter show dev enp7s0
filter parent 1: protocol ip pref 1 u32 chain 0
filter parent 1: protocol ip pref 1 u32 chain 0 fh 800: ht divisor 1
filter parent 1: protocol ip pref 1 u32 chain 0 fh 800::800 order 2048 key ht 800 bkt
0 flowid 1:11 not_in_hw
match c0a80004/ffffffff at 16
filter parent 1: protocol ip pref 1 u32 chain 0 fh 800::801 order 2049 key ht 800 bkt
0 flowid 1:12 not_in_hw
match c0a80700/ffffff00 at 16
```

## Wireless network setup

To set up a wireless network, you will need to:

- make sure that the wireless network card is technically capable of doing this;
- make sure that the wireless connection in the system is unlocked;
- configure the wireless access point and select its operation parameters;
- provide password protection for the access point;

- make sure that the access point is functioning and stable.

Each of these steps is described below.

## Managing a wireless network card

First of all, you should find out the technical capabilities of the wireless network card available in the system. To do this, just run the command:

```
root@server:/# iw list
Wiphy phy0
  max # scan SSIDs: 4
  max scan IEs length: 2257 bytes
  max # sched scan SSIDs: 0
  max # match sets: 0
  max # scan plans: 1
  max scan plan interval: -1
  max scan plan iterations: 0
  Retry short limit: 7
  Retry long limit: 4
  Coverage class: 0 (up to 0m)
  Device supports RSN-IBSS.
  Device supports AP-side u-APSD.
  Device supports T-DLS.
  Supported Ciphers:
  * WEP40 (00-0f-ac:1)
  * WEP104 (00-0f-ac:5)
  * TKIP (00-0f-ac:2)
  * CCMP-128 (00-0f-ac:4)
  * CCMP-256 (00-0f-ac:10)
  * GCMP-128 (00-0f-ac:8)
  * GCMP-256 (00-0f-ac:9)
  * CMAC (00-0f-ac:6)
  * CMAC-256 (00-0f-ac:13)
  * GMAC-128 (00-0f-ac:11)
  * GMAC-256 (00-0f-ac:12)
  Available Antennas: TX 0x1 RX 0x3
  Configured Antennas: TX 0x1 RX 0x3
  Supported interface modes:
  * IBSS
  * managed
  * AP
  * AP/VLAN
  * monitor
  * mesh point
  * P2P-client
  * P2P-GO
  * outside context of a BSS
  Band 1:
  Capabilities: 0x11ce
  HT20/HT40
  SM Power Save disabled
  RX HT40 SGI
  TX STBC
  RX STBC 1-stream
  Max AMSDU length: 3839 bytes
  DSSS/CCK HT40
    Maximum RX AMPDU length 65535 bytes (exponent: 0x003)
    Minimum RX AMPDU time spacing: 8 usec (0x06)
    HT TX/RX MCS rate indexes supported: 0-7
    Bitrates (non-HT):
```

- \* 1.0 Mbps
- \* 2.0 Mbps (short preamble supported)
- \* 5.5 Mbps (short preamble supported)
- \* 11.0 Mbps (short preamble supported)
- \* 6.0 Mbps
- \* 9.0 Mbps
- \* 12.0 Mbps
- \* 18.0 Mbps
- \* 24.0 Mbps
- \* 36.0 Mbps
- \* 48.0 Mbps
- \* 54.0 Mbps

Frequencies:

- \* 2412 MHz [1] (17.0 dBm)
- \* 2417 MHz [2] (17.0 dBm)
- \* 2422 MHz [3] (17.0 dBm)
- \* 2427 MHz [4] (17.0 dBm)
- \* 2432 MHz [5] (17.0 dBm)
- \* 2437 MHz [6] (17.0 dBm)
- \* 2442 MHz [7] (17.0 dBm)
- \* 2447 MHz [8] (17.0 dBm)
- \* 2452 MHz [9] (17.0 dBm)
- \* 2457 MHz [10] (17.0 dBm)
- \* 2462 MHz [11] (17.0 dBm)
- \* 2467 MHz [12] (17.0 dBm) (no IR)
- \* 2472 MHz [13] (17.0 dBm) (no IR)
- \* 2484 MHz [14] (disabled)

Supported commands:

- \* new\_interface
- \* set\_interface
- \* new\_key
- \* start\_ap
- \* new\_station
- \* new\_mpath
- \* set\_mesh\_config
- \* set\_bss
- \* authenticate
- \* associate
- \* deauthenticate
- \* disassociate
- \* join\_ibss
- \* join\_mesh
- \* remain\_on\_channel
- \* set\_tx\_bitrate\_mask
- \* frame
- \* frame\_wait\_cancel
- \* set\_wiphy\_netns
- \* set\_channel
- \* set\_wds\_peer
- \* tdls\_mgmt
- \* tdls\_oper
- \* probe\_client
- \* set\_noack\_map
- \* register\_beacons
- \* start\_p2p\_device
- \* set\_mcast\_rate
- \* connect
- \* disconnect
- \* channel\_switch
- \* set\_qos\_map
- \* set\_multicast\_to\_unicast

Supported TX frame types:

```

* IBSS: 0x00 0x10 0x20 0x30 0x40 0x50 0x60 0x70 0x80 0x90 0xa0 0xb0
0xc0 0xd0 0xe0 0xf0
* managed: 0x00 0x10 0x20 0x30 0x40 0x50 0x60 0x70 0x80 0x90 0xa0
0xb0 0xc0 0xd0 0xe0 0xf0
* AP: 0x00 0x10 0x20 0x30 0x40 0x50 0x60 0x70 0x80 0x90 0xa0 0xb0
0xc0 0xd0 0xe0 0xf0
* AP/VLAN: 0x00 0x10 0x20 0x30 0x40 0x50 0x60 0x70 0x80 0x90 0xa0
0xb0 0xc0 0xd0 0xe0 0xf0
* mesh point: 0x00 0x10 0x20 0x30 0x40 0x50 0x60 0x70 0x80 0x90 0xa0
0xb0 0xc0 0xd0 0xe0 0xf0
* P2P-client: 0x00 0x10 0x20 0x30 0x40 0x50 0x60 0x70 0x80 0x90 0xa0
0xb0 0xc0 0xd0 0xe0 0xf0
* P2P-GO: 0x00 0x10 0x20 0x30 0x40 0x50 0x60 0x70 0x80 0x90 0xa0 0xb0
0xc0 0xd0 0xe0 0xf0
* P2P-device: 0x00 0x10 0x20 0x30 0x40 0x50 0x60 0x70 0x80 0x90 0xa0
0xb0 0xc0 0xd0 0xe0 0xf0
Supported RX frame types:
* IBSS: 0x40 0xb0 0xc0 0xd0
* managed: 0x40 0xd0
* AP: 0x00 0x20 0x40 0xa0 0xb0 0xc0 0xd0
* AP/VLAN: 0x00 0x20 0x40 0xa0 0xb0 0xc0 0xd0
* mesh point: 0xb0 0xc0 0xd0
* P2P-client: 0x40 0xd0
* P2P-GO: 0x00 0x20 0x40 0xa0 0xb0 0xc0 0xd0
* P2P-device: 0x40 0xd0
software interface modes (can always be added):
* AP/VLAN
* monitor
valid interface combinations:
* #{ managed } <= 2048, #{ AP, mesh point } <= 8, #{ P2P-client, P2P-
GO } <= 1,
    total <= 2048, #channels <= 1, STA/AP BI must match
HT Capability overrides:
* MCS: ff ff ff ff ff ff ff ff ff ff
* maximum A-MSDU length
* supported channel width
* short GI for 40 MHz
* max A-MPDU length exponent
* min MPDU start spacing
Device supports TX status socket option.
Device supports HT-IBSS.
Device supports SAE with AUTHENTICATE command
Device supports low priority scan.
Device supports scan flush.
Device supports AP scan.
Device supports per-vif TX power setting
P2P GO supports CT window setting
Driver supports full state transitions for AP/GO clients
Driver supports a userspace MPM
Device supports active monitor (which will ACK incoming frames)
Driver/device bandwidth changes during BSS lifetime (AP/GO mode)
Device supports configuring vdev MAC-addr on create.
Supported extended features:
* [ RRM ]: RRM
* [ FILS_STA ]: STA FILS (Fast Initial Link Setup)
* [ CQM_RSSI_LIST ]: multiple CQM_RSSI_THOLD records
* [ CONTROL_PORT_OVER_NL80211 ]: control port over nl80211
* [ TXQS ]: FQ-CoDel-enabled intermediate TXQs

```

From the output of this command (the output was not shortened) it can be noted that the wifi card

- has the system name phy0;
- supports AP interface mode (Access Point);
- supports transfer speeds up to 54 Mbit / s (excluding HT capabilities);
- can use channels 1 to 13 inclusive.

The ability to work in access point mode is a necessary condition for the implementation of the task set for the system. Other parameters should be taken into account when configuring the software access point (hostapd daemon).

The rfkill utility can also be useful for enabling/disabling wireless network card, as well as for monitoring its status. To check the status of the card is sufficient to run this command without any parameters:

```
root@server:/# rfkill
ID TYPE DEVICE          SOFT      HARD
0 wlan ideapad_wlan    unblocked unblocked
1 wlan phy0            unblocked unblocked
```

The two devices shown are actually one physical device – the wireless network card specified in the description of the hardware component of the system. You can verify this by turning off this network card using a mechanical switch (HARD – "hard" shutdown) on the laptop body and performing it again:

```
root@server:/# rfkill
ID TYPE DEVICE          SOFT      HARD
0 wlan ideapad_wlan    unblocked blocked
1 wlan phy0            unblocked blocked
```

The same is observed when the device is locked using the keyboard (SOFT – "soft" shutdown):

```
root@server:/# rfkill
ID TYPE DEVICE          SOFT      HARD
0 wlan ideapad_wlan    blocked  unblocked
1 wlan phy0            blocked  blocked
```

This command, among other things, can be useful for remote unlocking of a wireless card if it is accidentally blocked by the user, of course, only in the case of "soft" blocking.

Obviously, before continuing to configure the system and creating an access point, make sure that the network card is unlocked.

## Configuring a software access point

The hostapd daemon is responsible for creating a software access point based on a configured wireless network card. First of all it must be installed in the system by the command:

```
root@server:/# apt-get install hostapd
```

Next, the configuration file of this daemon **/etc/hostapd** should be converted to the following format:

```
interface=wlp6s0
driver=nl80211
ssid="wifi at school34"
country_code=RU
hw_mode=g
ieee80211n=1
```



```
ht_capab=[HT40-][SHORT-GI-40]
channel=6
wpa=2
wpa_passphrase=48C74A6B
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
auth_algs=1
macaddr_acl=0
```

The following parameter values are specified in this file:

wpl6s0 – the wireless network interface of the system on which the access point should work.

nl80211 – network card driver to use. This driver is suitable for most network cards, including that used in the system.

ssid – the name of the access point in the form in which it is displayed for clients when connecting.

hw\_mode = g – network card operation mode (802.11 b/g/n). However, even if the card is capable of operating in 802.11n mode, you should still specify g mode here. Additional features introduced by the 802.11n standard are connected using the following two parameters.

ieee80211n – use 802.11n mode.

ht\_capab is a set of parameters for fine-tuning the wireless network parameters. The specific combination of these parameters is selected depending on the access point equipment, client device equipment, location of the access point on the ground, building architecture, the required coverage area, the presence of objects in the coverage area that can muffle or weaken the signal (mirrors, safes, microwave ovens, etc.), intersections with other wireless networks, and many other circumstances. The values that ensure acceptable quality of the wireless access point operation were selected experimentally on the system under consideration.

channel – used wireless channel, one of the default values (6) is quite suitable for use in the conditions under consideration.

The following parameters indicate how to protect your wireless network. This uses the Wi-Fi Protected Access 2 (WPA2) technology with a password permanently stored on the router itself (WPA-PSK). Without going into the technical details of WiFi, note the wpa\_passphrase parameter, since its value is the password for connecting to the access point. According to the logic of the described wireless network functioning within the premises where the wireless network server and client devices are installed, you should not make this password a secret. However, you need to change your password regularly to minimize the threat of connecting a client located outside of the specified premises. To solve this problem, we developed the command-line script described below, which changes the password in this file and restarts the service.

The last macaddr\_acl parameter, with the specified value, allows any client device to connect to the network.

Then, in the **/etc/default/hostapd** configuration file, uncomment the line

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

to ensure that the above configuration file is used and that the daemon is started at system startup.

## Automatic changing of the access point password

Automatic password changing is organized as follows: a script is created that generates a new password and places it in the required configuration files, this script is called when restarting the hostapd service, and the password is displayed on the laptop screen.

The script located in **/usr/local/sbin/wifipasswd** looks like:

```
#!/bin/bash

NEWPASSWORD=$(/usr/bin/makepasswd --string 0123456789ABCDEF --count 1 --chars 8)
/usr/bin/sed s/"^wpa_passphrase=.*$"/"wpa_passphrase=$NEWPASSWORD"/g -i /etc/hostapd/
hostapd.conf
echo -n $NEWPASSWORD > /usr/local/etc/wifi.password
exit 0
```

The script creates a new eight-character password consisting of hexadecimal digits (uppercase letters), which provides over 4 billion different combinations. Then the script changes the configuration file of the wireless access point so that it contains a new password, and also adds this new password to the file **/usr/local/etc/wifi.password**, which is displayed on the guest user's desktop using the RichTextViewer widget.

To run this scenario, you must add an extension to the hostapd service before each launch. To do this, create an **override.conf** file in the **/etc/systemd/system/hostapd.service.d** directory with the following contents:

```
[Service]
ExecStartPre=-/usr/local/sbin/wifipasswd
```

The location in the specified directory allows the file with service changes not to be erased during systemd updates. The best way to create such file is to use the command

```
root@server:/# systemctl edit hostapd
```

After you finish editing the service, restart it, check its status, and make sure that the changes are applied, for example, as shown in the following fragment of the terminal session:

```
root@server:/# systemctl restart hostapd
root@server:/# systemctl status hostapd
• hostapd.service - Advanced IEEE 802.11 AP and IEEE 802.1X/WPA/WPA2/EAP
Authenticator
   Loaded: loaded (/lib/systemd/system/hostapd.service; enabled; vendor preset:
enabled)
   Drop-In: /etc/systemd/system/hostapd.service.d
             └─override.conf
   Active: active (running) since Thu 2020-02-20 15:51:41 MSK; 3s ago
   Process: 2728 ExecStartPre=/usr/local/sbin/wifipasswd (code=exited,
status=0/SUCCESS)
   Process: 2731 ExecStart=/usr/sbin/hostapd -B -P /run/hostapd.pid -B $DAEMON_OPTS $
{DAEMON_CONF} (code=exited, status=0/SUCCESS)
   Main PID: 2732 (hostapd)
     Tasks: 1 (limit: 4509)
    Memory: 1.4M
    CGroup: /system.slice/hostapd.service
            └─2732 /usr/sbin/hostapd -B -P /run/hostapd.pid -B
/etc/hostapd/hostapd.conf
root@server:/# systemctl show hostapd
Type=forking
Restart=on-failure
```

```

PIDFile=/run/hostapd.pid
ExecMainStatus=0
ExecStartPre={ path=/usr/local/sbin/wifipasswd ; argv[]=/usr/local/sbin/wifipasswd ;
ignore_errors=yes ; start_time=[Thu 2020-02-20 15:51:41 MSK] ; stop_time=[Thu 2020-0
ExecStart={ path=/usr/sbin/hostapd ; argv[]=/usr/sbin/hostapd -B -P /run/hostapd.pid
-B $DAEMON_OPTS ${DAEMON_CONF} ; ignore_errors=no ; start_time=[Thu 2020-02-20
15:51:41 MS
Slice=system.slice
ControlGroup=/system.slice/hostapd.service
MemoryCurrent=1495040

```

The output of commands, which is insignificant in this context, is shortened here.

## Fail protection (auto-resume)

As you can see from the above fragment of the terminal session, the automatic restart of the service when it fails was already provided as part of the basic installation of the service (Restart=on-failure).

## Verification of performance and stability

After completing all settings, restart the service and check its status:

```

root@server:/# systemctl status hostapd
• hostapd.service - Advanced IEEE 802.11 AP and IEEE 802.1X/WPA/WPA2/EAP
Authenticator
  Loaded: loaded (/lib/systemd/system/hostapd.service; enabled; vendor preset:
enabled)
  Drop-In: /etc/systemd/system/hostapd.service.d
           └─override.conf
  Active: active (running) since Wed 2020-03-18 13:41:34 MSK; 25min ago
  Process: 644 ExecStartPre=/usr/local/sbin/wifipasswd (code=exited, status=0/SUCCESS)

  Process: 712 ExecStart=/usr/sbin/hostapd -B -P /run/hostapd.pid -B $DAEMON_OPTS $
{DAEMON_CONF} (code=exited, status=0/SUCCESS)
Main PID: 716 (hostapd)
  Tasks: 1 (limit: 4509)
  Memory: 7.2M
  CGroup: /system.slice/hostapd.service
          └─716 /usr/sbin/hostapd -B -P /run/hostapd.pid -B /etc/hostapd/hostapd.conf

```

As you can see from the command output, the process is running, active, and has PID 716.

Now you need to check the response of the service to the forced termination of the main process, "soft" and "hard" disconnection of the wireless adapter. However, network availability should be monitored from the client device. The following is a fragment of the terminal session that reflects the progress of such verification, a description of the verification process, and explanatory comments.

Fragment of a terminal session (with some minor abbreviations) :

```

root@server:/# kill -9 716
root@server:/# systemctl status hostapd
• hostapd.service - Advanced IEEE 802.11 AP and IEEE 802.1X/WPA/WPA2/EAP
Authenticator
  Loaded: loaded (/lib/systemd/system/hostapd.service; enabled; vendor preset:
enabled)
  Drop-In: /etc/systemd/system/hostapd.service.d
           └─override.conf

```

```

Active: active (running) since Wed 2020-03-18 14:15:45 MSK; 1s ago
Process: 1234 ExecStartPre=/usr/local/sbin/wifipasswd (code=exited,
status=0/SUCCESS)
Process: 1237 ExecStart=/usr/sbin/hostapd -B -P /run/hostapd.pid -B $DAEMON_OPTS $
{DAEMON_CONF} (code=exited, status=0/SUCCESS)
Main PID: 1238 (hostapd)
Tasks: 1 (limit: 4509)
Memory: 1.4M
CGroup: /system.slice/hostapd.service
└─1238 /usr/sbin/hostapd -B -P /run/hostapd.pid -B
/etc/hostapd/hostapd.conf
root@server:~# rfkill
ID TYPE DEVICE          SOFT    HARD
0 wlan ideapad_wlan    unblocked blocked
1 wlan phy0            unblocked blocked
root@server:~# systemctl status hostapd
• hostapd.service - Advanced IEEE 802.11 AP and IEEE 802.1X/WPA/WPA2/EAP
Authenticator
Loaded: loaded (/lib/systemd/system/hostapd.service; enabled; vendor preset:
enabled)
Drop-In: /etc/systemd/system/hostapd.service.d
└─override.conf
Active: active (running) since Wed 2020-03-18 14:15:45 MSK; 1min 52s ago
Process: 1234 ExecStartPre=/usr/local/sbin/wifipasswd (code=exited,
status=0/SUCCESS)
Process: 1237 ExecStart=/usr/sbin/hostapd -B -P /run/hostapd.pid -B $DAEMON_OPTS $
{DAEMON_CONF} (code=exited, status=0/SUCCESS)
Main PID: 1238 (hostapd)
Tasks: 1 (limit: 4509)
Memory: 1.4M
CGroup: /system.slice/hostapd.service
└─1238 /usr/sbin/hostapd -B -P /run/hostapd.pid -B
/etc/hostapd/hostapd.conf
root@server:~# rfkill
ID TYPE DEVICE          SOFT    HARD
0 wlan ideapad_wlan    unblocked unblocked
1 wlan phy0            unblocked unblocked
root@server:~# rfkill
ID TYPE DEVICE          SOFT    HARD
0 wlan ideapad_wlan    blocked  unblocked
1 wlan phy0            blocked  blocked
root@server:~# rfkill
ID TYPE DEVICE          SOFT    HARD
0 wlan ideapad_wlan    unblocked unblocked
1 wlan phy0            unblocked unblocked
root@server:~# systemctl status hostapd
• hostapd.service - Advanced IEEE 802.11 AP and IEEE 802.1X/WPA/WPA2/EAP
Authenticator
Loaded: loaded (/lib/systemd/system/hostapd.service; enabled; vendor preset:
enabled)
Drop-In: /etc/systemd/system/hostapd.service.d
└─override.conf
Active: active (running) since Wed 2020-03-18 14:15:45 MSK; 3min 17s ago
Process: 1234 ExecStartPre=/usr/local/sbin/wifipasswd (code=exited,
status=0/SUCCESS)
Process: 1237 ExecStart=/usr/sbin/hostapd -B -P /run/hostapd.pid -B $DAEMON_OPTS $
{DAEMON_CONF} (code=exited, status=0/SUCCESS)
Main PID: 1238 (hostapd)
Tasks: 1 (limit: 4509)
Memory: 1.4M
CGroup: /system.slice/hostapd.service
└─1238 /usr/sbin/hostapd -B -P /run/hostapd.pid -B

```

```
/etc/hostapd/hostapd.conf  
root@server:/#
```

Verification started with the client machine running continuous ping requests to the main server of the organization, the path to which is through the system being checked. Then the hostapd process is forcibly terminated. This is followed by a health check of the service, which shows that the service has already been restarted and is working with the new pid and password of the wireless network. This forces the client to delay sending ping packets and request a new password, after which sending requests resumes.

Then the "hard" disconnection of the wireless network card is performed by a mechanical switch on the device body and the status of this adapter is checked using the rfkill utility. The card is blocked, which causes the client to send messages to the terminal about network unavailability. Then the wireless adapter is unlocked in the same way, and its status is checked using rfkill again. The client resumes sending requests.

After that, the network card is "softly" disabled using the device's keyboard and its status is checked using rfkill. The client starts sending messages about network unavailability again. After enabling the network card and checking its status, sending requests on the client device resumes.

A second check of the service status indicates that there were no failures or restarts of the hostapd service during the time of disconnecting and enabling the wireless adapter.

Thus, we can conclude that the short-term lock of network card does not affect service health, and emergency restart of the service leads to changing the password to access the wireless network, which can be considered as an additional measure of system protection in the case that this failure was caused by malicious acts of some customer (e.g. in case of vulnerabilities in hostapd). In this case, the attacker will have to restore the network connection before repeating the attack.

## DNS/DHCP service

### Configuration

To implement this service, install the dnsmasq package with all dependencies:

```
root@server:/# apt-get update  
root@server:/# apt-get install dnsmasq
```

Then you should make the **/etc/dnsmasq.conf** configuration file look like this:

```
listen-address=127.0.0.1  
listen-address=192.168.4.1  
cache-size=500  
domain-needed  
bind-dynamic  
filterwin2k  
no-resolv  
no-poll  
no-hosts  
bogus-priv  
  
server=77.88.8.7@enp7s0  
server=/school34/192.168.0.1@enp7s0  
server=/0.168.192.in-addr.arpa/192.168.0.1@enp7s0
```

```
server=/service.school34/192.168.0.77@enp7s0
server=/7.168.192.in-addr.arpa/192.168.0.77@enp7s0

local=/wifi.school34/
domain=school34,192.168.0.0/24
domain=service.school34,192.168.7.0/24
domain=wifi.school34,192.168.4.0/24

mx-target=mail.service.school34
localmx

no-dhcp-interface=lo
no-dhcp-interface=enp7s0
dhcp-range=192.168.4.11,192.168.4.55,12h
dhcp-option=6,192.168.4.1
dhcp-option=42,192.168.4.1

#log-queries
#log-facility=/var/log/dnsmasq.log
```

Thus, the dnsmasq service is assigned the following operating procedure:

The service accepts requests only on the local loop and internal client interfaces. The cache of DNS records has been increased from 150 (by default) to 500 records, because the amount of RAM in the system does not prevent this, and this increase has a positive effect on the performance of the service. The volume of 500 records is selected as the most optimal based on the operating experience of a network, of which the wireless network created by the described system is a part.

The bind-dynamic parameter instructs the service to respond to changes in the state of network interfaces. This way, you can start the service when the device's wireless network interface is turned off and automatically "pick up" this interface when it is started. This approach allows you to prevent the service from crashing when you physically disable the wireless adapter in the way described in the hardware section.

The service does not use the system configuration files of the built-in DNS client. All the necessary parameters for its operation are specified in the main configuration file of the service, shown above.

The service knows three local domains (school34, service.school34, and wifi.school34). Each of them (except for the wifi.school34 domain served by the system itself) has its own DNS server, which is accessed via the external interface of the system. The bogus-priv parameter blocks DNS requests for other addresses that are not routed on the Internet. DNS queries about external hosts are resolved using YandexDNS, which can be easily replaced with a stricter one, as described in the section "Using the server".

The email service parameters are set so that the default email address for all wireless network clients is the address of the organization's local mail server (mail.service.school34).

The DHCP Protocol in the wireless subnet distributes addresses to clients in the range 192.168.4.11 – 192.168.4.55 (45 addresses, respectively, not more than 45 clients at a time). Requests to get an address are obviously only accepted from the system's wireless interface. In addition to providing the address, the server provides the client with DNS service (dhcp-option 6) and network time protocol service (dhcp-option 42).

The service keeps its log in the **/var/log/daemon.log** file, which briefly displays information about starting the service, its settings, and so on. Detailed service logging is disabled in this configuration, but you can enable logging by commenting out the last two lines in the file. At the same time, keep in mind that logging, even with the rotation of log files (see "About additional services"), can serve as a direction for attacking the server: sending a huge number of meaningless requests can overflow the server's disk space. In the system under consideration, there is no need for detailed logging of what is happening, so it is disabled.

After editing the configuration file, you can restart the service with the command

```
root@server:/# systemctl restart dnsmasq
```

To make sure that the restart is successful, follow the command output

```
root@server:/# systemctl status dnsmasq
```

```
• dnsmasq.service - dnsmasq - A lightweight DHCP and caching DNS server
  Loaded: loaded (/lib/systemd/system/dnsmasq.service; enabled; vendor preset:
  enabled)
  Active: active (running) since Wed 2020-01-29 10:23:28 MSK; 28s ago
  Process: 1860 ExecStartPre=/usr/sbin/dnsmasq --test (code=exited, status=0/SUCCESS)
  Process: 1861 ExecStart=/etc/init.d/dnsmasq systemd-exec (code=exited,
  status=0/SUCCESS)
  Process: 1870 ExecStartPost=/etc/init.d/dnsmasq systemd-start-resolvconf
  (code=exited, status=0/SUCCESS)
  Main PID: 1869 (dnsmasq)
    Tasks: 1 (limit: 4509)
    Memory: 1.7M
    CGroup: /system.slice/dnsmasq.service
            └─1869 /usr/sbin/dnsmasq -x /run/dnsmasq/dnsmasq.pid -u dnsmasq -7
  /etc/dnsmasq.d,.dpkg-dist,.dpkg-old,.dpkg-new --local-s

янв 29 10:23:28 server dnsmasq[1869]: compile time options: IPv6 GNU-getopt DBus i18n
IDN DHCP DHCPv6 no-Lua TFTP conntrack ipset aut
янв 29 10:23:28 server dnsmasq-dhcp[1869]: DHCP, IP range 192.168.4.11 --
192.168.4.55, lease time 12h
янв 29 10:23:28 server dnsmasq[1869]: using local addresses only for domain
wifi.school34
янв 29 10:23:28 server dnsmasq[1869]: using nameserver 192.168.0.77#53 for domain
7.168.192.in-addr.arpa
янв 29 10:23:28 server dnsmasq[1869]: using nameserver 192.168.0.77#53 for domain
service.school34
янв 29 10:23:28 server dnsmasq[1869]: using nameserver 192.168.0.1#53 for domain
0.168.192.in-addr.arpa
янв 29 10:23:28 server dnsmasq[1869]: using nameserver 192.168.0.1#53 for domain
school34
янв 29 10:23:28 server dnsmasq[1869]: using nameserver 77.88.8.7#53(via enp7s0)
янв 29 10:23:28 server dnsmasq[1869]: cleared cache
янв 29 10:23:28 server systemd[1]: Started dnsmasq - A lightweight DHCP and caching
DNS server.
```

The service runs on behalf of the dnsmasq system user. The process ID (PID) is entered in the **/run/dnsmasq/dnsmasq.pid** file by default. During operation, the dnsmasq process responds to the SIGUSR1 signal by sending messages to the system log about current clients, their request statistics, and so on. The following command allows you to send this signal to the desired process and open a fragment from the system log containing all mentions of dnsmasq for the last hour:

```
root@server:/# kill -s SIGUSR1 $(cat /run/dnsmasq/dnsmasq.pid) && journalctl -u
dnsmasq --since -1h
```

This sample has a form similar to:

```
-- Logs begin at Fri 2020-02-28 08:55:10 MSK, end at Fri 2020-02-28 15:17:47 MSK. --
фев 28 14:48:49 server dnsmasq-dhcp[766]: DHCPDISCOVER(wlp6s0) b8:8d:12:41:29:a2
фев 28 14:48:49 server dnsmasq-dhcp[766]: DHCPOFFER(wlp6s0) 192.168.4.21
b8:8d:12:41:29:a2
фев 28 14:48:49 server dnsmasq-dhcp[766]: DHCPDISCOVER(wlp6s0) b8:8d:12:41:29:a2
фев 28 14:48:49 server dnsmasq-dhcp[766]: DHCPOFFER(wlp6s0) 192.168.4.21
b8:8d:12:41:29:a2
фев 28 14:48:50 server dnsmasq-dhcp[766]: DHCPREQUEST(wlp6s0) 192.168.4.21
b8:8d:12:41:29:a2
фев 28 14:48:50 server dnsmasq-dhcp[766]: DHCPACK(wlp6s0) 192.168.4.21
b8:8d:12:41:29:a2
фев 28 14:53:11 server dnsmasq-dhcp[766]: DHCPREQUEST(wlp6s0) 192.168.4.16
d0:df:9a:6a:4c:b4
фев 28 14:53:11 server dnsmasq-dhcp[766]: DHCPACK(wlp6s0) 192.168.4.16
d0:df:9a:6a:4c:b4 server
фев 28 14:53:47 server dnsmasq-dhcp[766]: DHCPREQUEST(wlp6s0) 192.168.4.16
d0:df:9a:6a:4c:b4
фев 28 14:53:47 server dnsmasq-dhcp[766]: DHCPACK(wlp6s0) 192.168.4.16
d0:df:9a:6a:4c:b4 server
фев 28 15:17:47 server dnsmasq[766]: time 1582892267
фев 28 15:17:47 server dnsmasq[766]: cache size 500, 0/737 cache insertions re-used
unexpired cache entries.
фев 28 15:17:47 server dnsmasq[766]: queries forwarded 268, queries answered locally
132
фев 28 15:17:47 server dnsmasq[766]: queries for authoritative zones 0
фев 28 15:17:47 server dnsmasq[766]: server 192.168.0.77#53: queries sent 0, retried
or failed 0
фев 28 15:17:47 server dnsmasq[766]: server 192.168.0.1#53: queries sent 3, retried
or failed 0
фев 28 15:17:47 server dnsmasq[766]: server 77.88.8.7#53: queries sent 265, retried
or failed 0
```

When you finish configuring the service, you should tell the system to use its own service for its own needs. To do this, change the configuration file **/etc/resolv.conf** to the following:

```
nameserver 127.0.0.1
```

Then restart the network subsystem with the command:

```
root@server:/# systemctl restart networking
```

This is required so that dns queries generated by the system itself about local servers are sent directly to the virtualization server, rather than being redirected there by the main gateway.

## Fail protection (auto-resume)

You can use the systemd initialization tools to automatically restore the service after a failure.

First of all, you should check the current parameters of the service with the command:

```
root@server:/# systemctl show dnsmasq
Type=forking
Restart=no
PIDFile=/run/dnsmasq/dnsmasq.pid
NotifyAccess=none
RestartUsec=100ms
TimeoutStartUsec=1min 30s
TimeoutStopUsec=1min 30s
```



As you can see from the command response (it is shown here in a highly abbreviated form, containing important lines in the current context), restarting the service is disabled.

Then use the command to edit the service:

```
root@server:/# systemctl edit dnsmasq
```

This opens a text editor where you need to make certain edits. If you confirm that changes are saved when you exit the text editor, the system creates a file `/etc/systemd/system/dnsmasq.service.d/override.conf`, where these changes are stored. When the service is started again, they override the original parameter values. To solve the problem of restarting the service in case of failures, the following contents of this file are sufficient:

```
[Service]
Restart=on-failure
```

Such changes in the service are saved when installing system updates.

After making changes, you should restart the service and make sure that the parameter changes are taken into account:

```
root@server:/# systemctl restart dnsmasq
root@server:/# systemctl show dnsmasq
Type=forking
Restart=on-failure
PIDFile=/run/dnsmasq/dnsmasq.pid
NotifyAccess=none
RestartUsec=100ms
TimeoutStartUsec=1min 30s
TimeoutStopUsec=1min 30s
```

## Verification of performance and stability

Checking the health of the service is trivial: the client is connected via the DHCP Protocol and a number of DNS requests are made on it. The following fragment of a terminal session performed on a client device running the Xubuntu 19.04 live disk demonstrates that you are connecting to a wireless network, getting an ip address, network mask, network routes, and gateway and DNS server addresses (which are expected to match). The ability of the wireless network server to respond to DNS queries is then demonstrated:

```
xubuntu@xubuntu:~$ ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp13s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN
group default qlen 1000
    link/ether 54:be:f7:6f:8f:33 brd ff:ff:ff:ff:ff:ff
3: wlp14s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default qlen 1000
    link/ether 80:56:f2:08:71:7b brd ff:ff:ff:ff:ff:ff
    inet 192.168.4.34/24 brd 192.168.4.255 scope global dynamic wlp14s0
        valid_lft 43174sec preferred_lft 43174sec
    inet6 fe80::364a:99ac:fb9a:e6c4/64 scope link
        valid_lft forever preferred_lft forever
```

```

xubuntu@xubuntu:~$ ip route show
default via 192.168.4.1 dev wlp14s0 proto static metric 600
169.254.0.0/16 dev wlp14s0 scope link metric 1000
192.168.4.0/24 dev wlp14s0 proto kernel scope link src 192.168.4.34 metric 600
xubuntu@xubuntu:~$ systemd-resolve --status
Link 3 (wlp14s0)
    Current Scopes: DNS LLMNR/IPv4 LLMNR/IPv6
    LLMNR setting: yes
MulticastDNS setting: no
    DNSSEC setting: no
    DNSSEC supported: no
    DNS Servers: 192.168.4.1
    DNS Domain: wifi.school34
xubuntu@xubuntu:~$ host avers.school34
avers.school34 has address 192.168.0.4
xubuntu@xubuntu:~$ host library.service.school34
library.school34 has address 192.168.7.4
xubuntu@xubuntu:~$ host samba.service.school34
samba.school34 has address 192.168.7.2

```

In the presented fragment, some of the responses are abbreviated in the uninformative part.

Checking the system's stability to failures is demonstrated in the following fragment of the terminal session (responses of some commands are shortened in the non-formative part when applied to the context):

```

root@server:/# ps -ef | grep dnsmasq
dnsmasq  2061      1  0 15:18 ?        00:00:00 /usr/sbin/dnsmasq -x ...
root@server:/# kill -9 2061
root@server:/# ps -ef | grep dnsmasq
dnsmasq  2082      1  0 15:18 ?        00:00:00 /usr/sbin/dnsmasq -x
root@server:/# systemctl status dnsmasq
• dnsmasq.service - dnsmasq - A lightweight DHCP and caching DNS server
  Loaded: loaded (/lib/systemd/system/dnsmasq.service; enabled; vendor preset:
enabled)
  Drop-In: /etc/systemd/system/dnsmasq.service.d
           └─override.conf
  Active: active (running) since Fri 2020-02-07 15:18:52 MSK; 21s ago
  Process: 2073 ExecStartPre=/usr/sbin/dnsmasq --test (code=exited, status=0/SUCCESS)
  Process: 2074 ExecStart=/etc/init.d/dnsmasq systemd-exec (code=exited,

```

In this fragment, the PID of the current dnsmasq process (2061) is found out, then this process is intentionally terminated, and then the pid of the dnsmasq process is found out again. Because PID = 2082 exists and differs from the previous one, it is clear that the service was restarted after the crash. This is also confirmed by the output of the last command.

## Network time service (ntp)

### Configuration

First of all you need to install the ntp and ntpdate packages with all dependencies:

```

root@server:/# apt-get update
root@server:/# apt-get install ntp ntpdate

```

These packages include a number of programs:

- ntpd daemon,

- ntpq – standard program for queries,
- ntpdc – advanced program for queries,
- ntpdate – client program for setting time in the system over ntp,
- sntp – simple network client
- and others (key generators, service, debugging, simulators, etc.)

Before configuring **ntp.conf**, you should first configure the time zone (file **/etc/localtime**), which is best done using the command:

```
root@server:/# dpkg-reconfigure tzdata
```

To configure the exact time service, make the **/etc/ntp.conf** configuration file look like this:

```
# /etc/ntp.conf, configuration for ntpd; see ntp.conf(5) for help

driftfile /var/lib/ntp/ntp.drift

server ntp.school34      iburst prefer
server ntp1.vniiftri.ru iburst
server ntp2.vniiftri.ru iburst
server ntp3.vniiftri.ru iburst

interface ignore all
interface listen 192.168.0.44
interface listen 192.168.4.0/24

restrict default ignore
restrict ntp.school34      noquery notrap
restrict ntp1.vniiftri.ru noquery notrap
restrict ntp2.vniiftri.ru noquery notrap
restrict ntp3.vniiftri.ru noquery notrap
restrict 127.0.0.1
restrict ::1
restrict 192.168.4.0 mask 255.255.255.0 kod notrap nomodify nopeer noquery limited
```

Explanation of the configuration file:

**driftfile** – file for recording frequency correction of hardware clocks, updated by the daemon once an hour;

**interface** – a parameter that describes interaction with the system's network interfaces; the last appropriate one of several such directives is applied to the network packet.

**server** – an external server that is used for syncing, several such servers can be set in case one or more of them are unavailable, the additional parameter **prefer** means that in case of availability, use priority should be given to this server.

**Iburst** – send out 8 packets in 2 seconds instead of one, this allows syncing faster (in a few seconds instead of a few minutes), but is not suitable for networks with low bandwidth;

**kod** – kiss of death – send a response package with a notification in response to a package that violates the service load limits;

**notrap** – do not implement the functionality of determining the position of host in IPv6;

**nomodify** – reject requests that try to change the server state, only requests that only get a response are allowed;

nopeer – reject unauthorized requests to establish a connection, does not apply to packages that do not establish a connection, i.e. serve clients, but do not sync with them;

restrict default – this string specifies the default restrictions, here by default all packages except those explicitly marked below are ignored;

restrict – enter restrictions on the host or network (the host can be set as a name or address, it is pointless to set a host name that has multiple ip addresses, such as debian.pool.ntp.org, because in this case, the default rule will be triggered), the network is set by its address and mask, followed by keys that restrict this network or host;

limited – reject synchronization requests if the traffic limits set by the discard command are exceeded (the minimum interval between packets is 1 second by default, and the average interval is 3 seconds), if the kod flag is also set, a response packet is sent;

noquery – reject requests from ntpq and ntpdc, the exact time service is not affected;

Thus, the configuration file forces ntpd to work according to the following rules:

1. syncing is only allowed with pre-defined ntp servers
2. by default, all packets sent to the server are ignored, except those explicitly allowed
3. external ntp servers are not allowed to access the local server with service requests or use IPv6
4. all connections from the server itself are allowed, i.e. you can make any requests to the ntp server locally, manage it, and track its status
5. local clients are allowed only from the specified subnet, they are limited in bandwidth, can not influence the time server and send service requests to it, they are only allowed to get the exact time from the server

All that remains is to restart the exact time subsystem with the command

```
root@server:/# systemctl restart ntp
```

and make sure that the service is in a working state by using the command

```
root@server:/# systemctl status ntp
```

If there are no errors, the output of this command will be similar to:

```
• ntp.service - Network Time Service
Loaded: loaded (/lib/systemd/system/ntp.service; enabled; vendor preset: enabled)
Active: active (running) since Fri 2020-02-28 10:54:07 MSK; 3min 8s ago
Docs: man:ntpd(8)
Process: 1864 ExecStart=/usr/lib/ntp/ntp-systemd-wrapper (code=exited,
status=0/SUCCESS)
Main PID: 1870 (ntpd)
Tasks: 2 (limit: 4509)
Memory: 1.8M
CGroup: /system.slice/ntp.service
└─1870 /usr/sbin/ntpd -p /var/run/ntpd.pid -4 -g -u 122:127
```

```
Feb 28 10:54:07 server ntpd[1864]: Command line: /usr/sbin/ntpd -p /var/run/ntpd.pid
-4 -g -u 122:127
Feb 28 10:54:07 server systemd[1]: Started Network Time Service.
Feb 28 10:54:07 server ntpd[1870]: proto: precision = 0.175 usec (-22)
Feb 28 10:54:07 server ntpd[1870]: Listen and drop on 0 v4wildcard 0.0.0.0:123
Feb 28 10:54:07 server ntpd[1870]: Listen normally on 1 lo 127.0.0.1:123
Feb 28 10:54:07 server ntpd[1870]: Listen normally on 2 enp7s0 192.168.0.44:123
```

```

Feb 28 10:54:07 server ntpd[1870]: Listen normally on 3 wlp6s0 192.168.4.1:123
Feb 28 10:54:07 server ntpd[1870]: Listening on routing socket on fd #20 for
interface updates
Feb 28 10:54:07 server ntpd[1870]: kernel reports TIME_ERROR: 0x2041: Clock
Unsynchronized
Feb 28 10:54:07 server ntpd[1870]: kernel reports TIME_ERROR: 0x2041: Clock
Unsynchronized

```

In this response, you should note messages that the clock is not synchronized (Clock Unsynchronized), which is expected, because the service has just started, and synchronization takes some time. The following commands are executed two minutes after restarting the service, demonstrate that the synchronization is successful:

```

root@server:/# ntpdate -q localhost
server 127.0.0.1, stratum 2, offset 0.000001, delay 0.02568
28 Feb 10:57:21 ntpdate[1882]: adjust time server 127.0.0.1 offset 0.000001 sec
root@server:/# ntpq -p
remote          refid st t when poll reach  delay  offset jitter
=====
*ntp.school34    .MRS.  1 u   4   64   17 28.163  0.193  5.616
+ntp1.vniiftri.ru .MRS.  1 u   3   64   17 28.331  0.856  0.970
+ntp2.vniiftri.ru .MRS.  1 u   5   64   17 30.655  2.230  0.694

```

In particular, it stated that the system clock is synchronized to the local server with a precision of 0.000001 seconds, the local server has a stratum level of 2, the deviation of its clocks from upstream ntp server.school34 clocks is 0.193 milliseconds, and the variance of the deviations on the results of a few recent queries is 5.616 milliseconds.

With the specified settings, the ntpd daemon nevertheless listens to all system interfaces, but ignores packets according to its configuration file. The daemon also listens to IPv6 addresses. You can see this in the output of the command:

```

root@server:/# ss -ulp | grep ntp
UNCONN      0      0      192.168.4.1:ntp      0.0.0.0:*
UNCONN      0      0      192.168.0.44:ntp     0.0.0.0:*
UNCONN      0      0      127.0.0.1:ntp        0.0.0.0:*
UNCONN      0      0      0.0.0.0:ntp          0.0.0.0:*
UNCONN      0      0      [::]:ntp             [::]:*

```

To force the daemon to disable IPv6 support, make the **/etc/default/ntp** configuration file look like this:

```
NTPD_OPTS='-4 -g'
```

After restarting the service, you can make sure that IPv6 is no longer used:

```

root@server:/# systemctl restart ntp
root@server:/# ss -ul | grep ntp
UNCONN      0      0      192.168.4.1:ntp      0.0.0.0:*
UNCONN      0      0      192.168.0.44:ntp     0.0.0.0:*
UNCONN      0      0      127.0.0.1:ntp        0.0.0.0:*
UNCONN      0      0      0.0.0.0:ntp          0.0.0.0:*

```

The service keeps its system log in the file **/var/log/daemon.log**, where other services write their messages. To get a sample of messages related to the ntp service for a certain time interval (for example, 08:00-10:30 of the current day), use the command

```
root@server:/# journalctl -u ntp --since 08:00 --until 10:30
```

## Fail protection (auto-resume)

You can use the systemd initialization tools to automatically restore the service after a failure.

First of all, you should check the current parameters of the service with the command:

```
root@server:/# systemctl show ntp
Type=forking
Restart=no
NotifyAccess=none
RestartUsec=100ms
TimeoutStartUsec=1min 30s
TimeoutStopUsec=1min 30s
```

As you can see from the command response (it is shown here in a highly abbreviated form, containing important lines in the current context), restarting the service is disabled.

Then use the command to edit the service:

```
root@server:/# systemctl edit ntp
```

This opens a text editor where you need to make certain edits. If you confirm that the changes are saved when you exit the text editor, a file **/etc/systemd/system/ntp.service.d/override.conf** is created in the system, where these changes are stored. When the service is started again, they override the original parameter values. To solve the problem of restarting the service in case of failures, the following contents of this file are sufficient:

```
[Service]
Restart=on-failure
```

Such changes in the service are saved when installing system updates.

After making changes you should restart the service and make sure that the parameter changes are taken into account:

```
root@server:/# systemctl restart ntp
root@server:/# systemctl show ntp
Type=forking
Restart=on-failure
NotifyAccess=none
RestartUsec=100ms
TimeoutStartUsec=1min 30s
TimeoutStopUsec=1min 30s
```

## Verification of performance and stability

Checking the health of the service is trivial: a request for time correction is made from a client connected to a wireless network (in this case, a laptop running Debian 10 that has deliberately distorted the system time). These actions are reflected in the following fragment of the terminal session:

```
root@debian:/# date --set="16:03"
Пт фев 28 16:03:00 MSK 2020
root@debian:/# /sbin/ntpdate 192.168.4.1
28 Feb 15:51:26 ntpdate[1393]: adjust time server 192.168.4.1 offset 0.000285 sec
```

Checking the system's stability to failures is demonstrated in the following fragment of the terminal session (responses of some commands are shortened in the non-formative part when applied to the context):

```
root@server:/# systemctl status ntp
• ntp.service - Network Time Service
  Active: active (running) since Fri 2020-02-28 11:29:19 MSK; 4h 29min ago
Main PID: 2049 (ntpd)
  Tasks: 2 (limit: 4509)
  Memory: 1.7M
  CGroup: /system.slice/ntp.service
          └─2049 /usr/sbin/ntpd -p /var/run/ntpd.pid -4 -g -u 122:127
root@server:/# kill -9 2049
root@server:/# systemctl status ntp
• ntp.service - Network Time Service
  Active: active (running) since Fri 2020-02-28 15:58:46 MSK; 3s ago
Main PID: 2522 (ntpd)
  Tasks: 2 (limit: 4509)
  Memory: 1.7M
  CGroup: /system.slice/ntp.service
          └─2522 /usr/sbin/ntpd -p /var/run/ntpd.pid -4 -g -u 122:127
```

In this fragment, the status of the exact time service is checked, the response message finds out the PID of the current ntpd process (2049), then this process is intentionally terminated, after which the service status is checked again and the PID of the newly started ntpd process (2522) is found out. The health of the service is also confirmed by repeated successful synchronization with the server of the same client.

## Installing and configuring ssh

Ssh installation can be performed both at the system installation stage and after using the commands

```
root@server:/# apt-get update
root@server:/# apt-get install ssh
```

On this server, ssh is used as the only remote login and administration tool. Taking into account the location of the server in the network, ssh accepts connections only on the wired (inaccessible to clients) interface. However, significant measures must be taken to protect the remote administration system. These measures are divided into three levels: the network filter, the basic configuration of the server and the configuration of the ssh service itself.

The configuration of network filtering is described below, and the basic configuration of the server is described above when describing access lists and static arp records.

Regarding the configuration of the service itself, two areas should be distinguished: protection from unauthorized access and ensuring trouble-free operation.

## Configuration

The main configuration file of the service **/etc/ssh/sshd\_config** should be made to correspond the following settings:

```
Port 22
ListenAddress 192.168.0.44
AddressFamily inet
```

```
Protocol 2
PermitRootLogin no
AllowUsers guest
PasswordAuthentication yes
PubkeyAuthentication no
KerberosAuthentication no
HostbasedAuthentication no
IgnoreRhosts yes
PermitEmptyPasswords no
X11Forwarding yes
```

This ensures that the ssh server works in the following order:

Port 22 – the server accepts messages on port 22 over the tcp protocol.

Protocol 2 – using ssh version 2.

AddressFamily inet – IPv4 is allowed, IPv6 is disabled.

PermitRootLogin no – remote login on behalf of the superuser is prohibited.

AllowUsers guest – remote login is only allowed for the guest user.

PasswordAuthentication yes – only password authentication is allowed, other methods are forbidden because they are not used.

PermitEmptyPasswords no – use of empty passwords is prohibited.

X11 forwarding yes – traffic of the X11 protocol is allowed, because it is used by the server.

After this file is edited, restart the ssh service, then make sure that it is started successfully and only accepts IPv4 connections. You can do this with the following commands:

```
root@debian:/# systemctl restart sshd
root@debian:/# systemctl status sshd
• ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2020-02-03 16:13:52 MSK; 8s ago
    Docs: man:sshd(8)
          man:sshd_config(5)
  Process: 1832 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 1833 (sshd)
   Tasks: 4 (limit: 4509)
  Memory: 7.4M
   CGroup: /system.slice/ssh.service
           └─1248 sshd: guest [priv]
             └─1304 sshd: guest@pts/0
               └─1307 -bash
                 └─1833 /usr/sbin/sshd -D
```

```
feb 03 16:13:52 server systemd[1]: This usually indicates unclean termination of a
previous run, or service implementation deficiencies.
```

```
feb 03 16:13:52 server systemd[1]: Starting OpenBSD Secure Shell server...
```

```
feb 03 16:13:52 server sshd[1833]: Server listening on 192.168.0.44 port 22.
```

```
feb 03 16:13:52 server systemd[1]: Started OpenBSD Secure Shell server.
```

```
root@debian:/# ss -Htulp | grep ssh
```

```
tcp LISTEN 0 128 192.168.0.44:ssh 0.0.0.0:* users: ("sshd",pid=1833,fd=3))
```

The service keeps its system log in the file **/var/log/daemon.log**, where other services write their messages. To get a selection of ssh-related messages for a certain time interval (for example, for the current day), use the command



```
root@server:/#journalctl -u ssh --since today
```

## Fail protection (auto-resume)

The system's services are managed by the systemd, which ensures that services automatically resume if they crash (if configured properly). In the system under consideration this configuration is performed initially by the OS distribution's assemblers.

## Verification of performance and stability

To check the service status, use the command

```
root@debian:/# systemctl status ssh
```

or a command

```
root@debian:/# service ssh status
```

which is, in fact, a wrapper of the above. On the system in question, due to the value of the PATH environment variable, the last command must be executed in the following form (if you do not modify the PATH value):

```
root@debian:/# /sbin/service ssh status
```

To check which ports, addresses and interfaces ssh is listening for, use the command

```
root@debian:/# ss -Htulp | grep ssh
```

The following fragment of the terminal session demonstrates attempts to log in remotely from an authorized host under different accounts:

```
administrator@admin:~$ ssh guest@server.wifi.school34
guest@server.wifi.school34's password:
Last login: Fri Feb 21 14:01:07 2020 from 192.168.0.2
```

The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/\*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

```
guest@server:~$ logout
Connection to server.wifi.school34 closed.
administrator@admin:~$ ssh root@server.wifi.school34
root@server.wifi.school34's password:
Permission denied, please try again.
root@server.wifi.school34's password:
Permission denied, please try again.
root@server.wifi.school34's password:
root@server.wifi.school34: Permission denied (password,keyboard-interactive).
administrator@admin:~$
```

As you can see from this snippet, the guest login was successful, and the root login was not performed even though the correct password was entered.

When trying to log in from a host other than the allowed one, the client receives a message (if the firewall of the server is disabled):

```
xubuntu@xubuntu:~$ ssh guest@server.wifi.school34
ssh_exchange_identification: read: Connection reset by peer
```

This is because of of access control lists. When the firewall is enabled the client receives a different message:

```
xubuntu@xubuntu:~$ ssh guest@server.wifi.school34
ssh: connect to host guest@server.wifi.school34 port 22: Connection timed out
```

However, in both cases connection to the server is denied.

One of the ways to check for automatic health restoration is shown in the following terminal session:

```
root@server:/# date
Пт фев 21 16:11:47 MSK 2020
root@server:/# ps -ef | grep ssh
root      708      1   0 11:14 ?        00:00:00 /usr/sbin/sshd -D
guest    1192    1159   0 11:14 ?        00:00:00 /usr/bin/ssh-agent /usr/bin/startkde
root     4075    4053   0 16:11 pts/2    00:00:00 grep ssh
root@server:/# kill -9 708
root@server:/# ps -ef | grep ssh
guest    1192    1159   0 11:14 ?        00:00:00 /usr/bin/ssh-agent /usr/bin/startkde
root     4083      1   0 16:12 ?        00:00:00 /usr/sbin/sshd -D
root     4086    4053   0 16:12 pts/2    00:00:00 grep ssh
root@server:/# systemctl status sshd
• ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2020-02-21 16:12:29 MSK; 15s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 4082 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 4083 (sshd)
    Tasks: 1 (limit: 4509)
   Memory: 1.1M
    CGroup: /system.slice/ssh.service
            └─4083 /usr/sbin/sshd -D

фев 21 16:12:29 server systemd[1]: Starting OpenBSD Secure Shell server...
фев 21 16:12:29 server sshd[4083]: Server listening on 192.168.0.44 port 22.
фев 21 16:12:29 server systemd[1]: Started OpenBSD Secure Shell server.
root@server:/#
```

Here, the system date and time are displayed first, then a list of running processes in the system is obtained, filtered by ssh mention. As you can see from this list, the system is running an sshd process with pid 708. The next command kills this process. It is further demonstrated that the process was actually killed, but a new process with PID 4083 was already started and the service was successfully restarted.

## About additional services

### Password protection (fail2ban)

Since ssh is the only remote login mechanism due to the nature of the server, it is quite safe to refuse to use fail2ban. The system is well protected by a firewall (described below) that specifies (both by ip and mac address) the only host from which SSH login is allowed. The ssh server of the system itself has the same restrictions, in addition, measures have been taken to protect against

arp attacks (static arp tables entries), remote login is allowed only for an unprivileged user whose account is protected by a password, and increasing privileges requires knowledge of the superuser's password.

## Log rotation (logrotate)

System logs can serve as a means of attacking the server in order to fill its disk space and thus paralyze its operation. This is opposed by the log file rotation system.

When logging in over ssh, the corresponding entry is entered in the **/var/log/auth.log** file. Taking into account the ssh protection measures taken, it can be argued that the attack is possible only from a single host that is not less protected from the network core (the system administrator's computer).

The other services available to clients (dnsmasq and ntp) do not maintain their own logs of client requests and thus cannot be targeted for such attack.

As a result, it should be noted that the default system settings (rotation weekly) are quite applicable here.

## Network time service (ntp)

The exact time service server not only provides services to clients, but also monitors the accuracy of the system clock of the machine on which it is deployed. Therefore, it is impossible and pointless to work with other clock synchronization tools in the described system. In particular, the simplest ftp client, which is part of systemd, turns out to be inoperable after installing and configuring ntpd and must be disabled:

```
root@server:/# systemctl status systemd-timesyncd
• systemd-timesyncd.service - Network Time Synchronization
  Loaded: loaded (/lib/systemd/system/systemd-timesyncd.service; enabled; vendor
  preset: enabled)
  Drop-In: /usr/lib/systemd/system/systemd-timesyncd.service.d
           └─disable-with-time-daemon.conf
  Active: inactive (dead)
  Condition: start condition failed at Mon 2020-03-02 10:09:05 MSK; 15min ago
             └─ ConditionFileIsExecutable=!/usr/sbin/ntpd was not met
  Docs: man:systemd-timesyncd.service(8)
```

```
map 02 10:09:05 server systemd[1]: Condition check resulted in Network Time
Synchronization being skipped.
```

```
root@server:/# systemctl disable systemd-timesyncd
Removed /etc/systemd/system/dbus-org.freedesktop.timesync1.service.
Removed /etc/systemd/system/sysinit.target.wants/systemd-timesyncd.service.
```

## Mail service

A mail service, even a local one, can become a target for an attack: for example, an attacker can generate a huge volume of emails locally in the system, causing file system overflow and service failure. Therefore, even the local mail service should be carefully configured or disabled if it is not used at all.

In this system, along with other components of the desktop environment, the mail transfer agent (MTA) exim4 is installed. In the future, it is planned to put into operation the internal mail server of the organization, which will allow many system services (for example, the hard disk health

monitoring system described below) to notify the system administrator about problems. However, users are not supposed to use MTA: it is much more reasonable and practical to use the web interface of the organization's mail server on a shared machine.

From the point of view of security MTA can be used to overflow user mailboxes, i.e. fill the disk space of the machine.

At this stage, it makes sense to completely remove the MTA from the system, especially since this does not lead to the removal of other system components. So, the following fragment of the terminal session, abbreviated in insignificant output, demonstrates the procedure for checking the possibility of deleting and deleting the MTA itself:

```
root@server:/# dpkg -l | grep exim
ii exim4-base 4.92-8+deb10u3 amd64 support files for all Exim MTA (v4) packages
ii exim4-config 4.92-8+deb10u3 all configuration for the Exim MTA (v4)
ii exim4-daemon-light 4.92-8+deb10u3 amd64 lightweight Exim MTA (v4) daemon
root@server:/# apt-get -s purge exim4-daemon-light
Чтение списков пакетов... Готово
Построение дерева зависимостей
Чтение информации о состоянии... Готово
Следующие пакеты устанавливались автоматически и больше не требуются:
 exim4-base exim4-config keyutils libgnutls-dane0 libnfsidmap2 libunbound8
Для их удаления используйте «apt autoremove».
Следующие пакеты будут УДАЛЕНЫ:
 exim4-daemon-light*
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 1 пакетов, и
117 пакетов не обновлено.
Purge exim4-daemon-light [4.92-8+deb10u3]
root@server:/# apt-get purge exim4-daemon-light
Чтение списков пакетов... Готово
...
root@server:/# apt autoremove
...
Следующие пакеты будут УДАЛЕНЫ:
 exim4-base exim4-config keyutils libgnutls-dane0 libnfsidmap2 libunbound8
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 6 пакетов, и
116 пакетов не обновлено.
...
root@server:/#
```

Since some system services use the mail system in their work, you should make sure that their functionality is not disrupted. So after a successful system reboot (in the sense of no error messages about starting services), you should check their status. On the system in question, such services are cron and smartd (the service is described below):

```
root@server:/# systemctl status cron
• cron.service - Regular background program processing daemon
  Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
  Active: active (running) since Fri 2020-03-13 16:50:49 MSK; 7min ago
...
map 13 16:51:15 server CRON[538]: (CRON) info (No MTA installed, discarding output)
...
root@server:/# systemctl status smartd
• smartd.service - Self Monitoring and Reporting Technology (SMART) Daemon
  Loaded: loaded (/lib/systemd/system/smartd.service; enabled; vendor preset:
enabled)
  Active: active (running) since Fri 2020-03-13 16:50:49 MSK; 7min ago
...

```

The output of commands that is not important in this context is also shortened here.

In the future, after the organization's internal mail server is deployed, MTA can again be installed in the system and configured accordingly.

## **Application software**

Despite the fact that the laptop in question acts as a wireless subnet server, it does not use most of its computing resources. In addition, the most logical solution for displaying a regularly changing wireless network password is to use the laptop screen. Therefore, it is possible to perform the role of a workstation. In this case it is necessary to ensure that the graphical working environment remains in an acceptable state, as well as to ensure that the system remains functional regardless of the actions of an unprivileged user. It is also necessary to place a sufficient set of application software in the system.

## **Application software Suite and graphical environment**

Plasma5 was chosen as the desktop environment to maintain commonality with other linux systems in the organization. To work as a workstation the following software is installed on the laptop among other things:

- LibreOffice 6.1 office suite;
- Calligra Suite 3.1.0 office suite;
- program for viewing pdf documents Okular 1.3.2;
- universal scanner program XSane 0.999;
- ocrfeeder 0.8.1 optical character recognition system;
- electronic dictionary GoldenDict 1.5.0;
- desktop publishing program Scribus 1.4.8;
- GIMP 2.10.8 raster graphics editor;
- vector graphics editor Inkscape 0.92;
- Mozilla Firefox 68 Internet browser;
- chromium 78 Internet browser;
- Audacity 2.2.2 sound editor;
- sound editor Kwave 18.08;
- vlc 3.0.8 media player;
- program for writing and copying optical disks K3b 18.08;
- client for connecting to the remote desktop KRDC 18.04.

A large set of auxiliary software tools is also installed: simple text editors, calculator, calendar, programs for capturing video from web camera and screen, a program for creating screenshots, and so on.

For ease of use, the graphical user interface has been configured, including:

- general interface configuration;
- the most optimal power supply plan is configured – system shutdown during idle time, switching to standby mode, screen blanking, etc. are disabled.
- shortcuts to the most frequently used programs are displayed on the taskbar;
- network service shortcuts (file server, AIS "Avers", etc.) are placed on the desktop;
- using the RichTextViewer widget the current wireless network password is displayed;
- the most frequently used programs were launched for the first time;

- automatic change of desktop wallpaper for the seasons and before holidays provided;
- the server automatically turns off at the end of the working day.

Some of these measures, which were not trivial to implement, are described below.

## Automatic restoration of interface and program settings

Automatic recovery of interface and program settings is implemented using a prepared archive with reference settings and a script designed to replace the current settings with those from the reference archive. While restoring settings, the local cache stored in the user's home directory, browser logs, and all guest files created by the user are also cleared. You should pay special attention to the latter – no documents created by the user, no downloads are saved after the script is running. This is done because the system works in guest mode and is not associated with any specific user. If the user name in the system is not guest, only the settings are restored without deleting files.

All this is implemented using the `restore-settings` script developed in-house. It is installed in the system as a package of the same name from the local repository. The script itself is located in the `/usr/bin/restore-settings` file, and detailed instructions for it are into the package as a man-page in Russian and English. Follow <https://sourceforge.net/projects/restoresettings/> to get the package .

The script is called every time the system is rebooted using the cron daemon, for which the following lines are added to its `/etc/crontab` configuration file:

```
#Restoring GUI settings, clearing cache, removing guest's files
@reboot root restore-settings guest /home/guest /opt/templates/guest.tar guest 0755
```

It should also be noted that this restoration of settings does not prevent you from displaying a changing wireless network password, or changing your desktop wallpaper. The fact is that only files in the guest user's home directory are subject to forced recovery, and there are only permanent links to really changing content. The file where the password is stored is described above in the section "Wireless network setup". As for the desktop wallpaper, the wallpaper file is `/usr/local/share/wallpapers/current.jpg` , which is a hard link to one of the images in the same directory. By changing this link on a schedule, you can achieve the desired effect. Only files stored in the KDE system cache (Plasma 5) can cause some difficulty, but it is sufficient to clear them at the same time as the link changes. To organize all of the above it is enough to place the following two files in the cron daemon tasks directory:

```
root@server:/# cat /etc/cron.d/wallpapers
#Changing wallpapers by seasons and holidays
* * 29-31 8 * root ln -f /usr/local/share/wallpapers/1september.jpg
    /usr/local/share/wallpapers/current.jpg
* * 1 9 * root ln -f /usr/local/share/wallpapers/1september.jpg
    /usr/local/share/wallpapers/current.jpg
root@server:/# cat /etc/cron.d/guest
#Changing wallpapers by seasons and holidays
* * 29-31 8 * root rm -rf /var/tmp/kdecache-guest
* * 1 9 * root rm -rf /var/tmp/kdecache-guest
```

Here both files are represented in abbreviated form by two lines each (in the first file, the lines are split into two due to the page format).

## Displaying the current password

As mentioned above, the current password for accessing the wireless network is displayed using the Plasma5 RichTextViewer widget, i.e. in a large font on a contrasting background directly on the desktop, which allows customers who are in the same room with the wireless network server to recognize it easily without unnecessary actions.

## Automatic system shutdown

Automatic shutdown of the system during non-working hours is also performed by means of the cron daemon, for which the following lines are added to its **/etc/crontab** configuration file:

```
#Shutting down at night
50 20 * * * root shutdown +10
```

They ensure that the system is shut down at 21: 00 with 10 minutes' notice to users. As a result, the next time you turn on your computer, you clear your home directory, restore settings, and change your wireless network password. Obviously, shutting down the system at night has three goals:

- energy saving;
- save the resource of the hardware platform;
- countering long-term attacks on password selection.

## Installing and configuring the network filter (nftables)

### Network environment and potential threats

Creating a set of network filter rules should start by finding out all the protocols and ports used by the system and clients. You can get a list of network ports and protocols on which the system is ready to accept connections as follows:

```
root@server:/tmp# ss -tuln
```

Netid	State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port
udp	UNCONN	0	0	127.0.0.1:53	0.0.0.0:*
udp	UNCONN	0	0	192.168.4.1:53	0.0.0.0:*
udp	UNCONN	0	0	0.0.0.0:67	0.0.0.0:*
udp	UNCONN	0	0	192.168.0.44:123	0.0.0.0:*
udp	UNCONN	0	0	192.168.4.1:123	0.0.0.0:*
udp	UNCONN	0	0	127.0.0.1:123	0.0.0.0:*
udp	UNCONN	0	0	0.0.0.0:123	0.0.0.0:*
udp	UNCONN	0	0	0.0.0.0:41410	0.0.0.0:*
udp	UNCONN	0	0	0.0.0.0:631	0.0.0.0:*
tcp	LISTEN	0	32	127.0.0.1:53	0.0.0.0:*
tcp	LISTEN	0	32	192.168.4.1:53	0.0.0.0:*
tcp	LISTEN	0	128	192.168.0.44:22	0.0.0.0:*
tcp	LISTEN	0	5	127.0.0.1:631	0.0.0.0:*

A number of observations should be made:

First, you should close by the network filter port 67 for UDP protocol for all incoming packets via the wired interface. This port is used by the DHCP protocol, which, according to the task statement, should only serve wireless network clients. This port is opened by the dnsmasq daemon, which also performs this check, but additional firewall protection is clearly not redundant.

Second, both interfaces need to close port 631 for tcp and udp at this stage. This port should remain available only for the local loop interface. Moreover, even if the system later implements a print server for wireless network clients, it is unnecessary to open the UDP port for both interfaces: the cups-browsed daemon accepts information about printers available on the network through it. But with given structure of the network, if they are implemented, information about them will be entered into the system on a permanent basis.

Third, it is not superfluous to close the wired interface for new connections over the network time protocol (port 123) without relying only on the stability of the ntpd daemon itself. However, the ability of the daemon itself to initiate connections to higher-level network time servers will not be affected.

Fourth, to reduce the likelihood of denial-of-service attacks, the maximum number of packets per unit of time for each service is limited.

Finally, you should provide maximum protection for the ssh service that is critical for system security, namely, allow connections to it only by checking the ip address, hardware address, interface from which the packet came, and observing the limit on the number of such packets per unit of time.

The issue of packet transit should be considered separately. To avoid attempts to circumvent the restrictions imposed on the wireless network by the task statement, you should leave the possibility of transit of network packets only to the AIS "AVERS" server (over tcp port 8082), to the virtual subnet of the organization's servers (without port restrictions at this stage) and to the Internet (only to ports from a certain list, later in this section they will be called "white").

However, it should be understood that the Internet refers to any network address other than the ones listed above, including addresses from other local subnets that the wireless network server does not know anything about. It would seem that an attacker, knowing the structure of the organization's network, can send a packet to one of these "white" ports to some computer, for example, in an administrative subnet. However, because the wireless network server does not have a route to this subnet in its routing table, and because it is not possible to direct the packet to the administrative subnet server via its address in the network core, the only route will be the main gateway, whose firewall will stop sending a dangerous packet. In addition, firewall protection is also organized at the server of the attacked subnet.

However, even with this organization of the network filter, you can not be sure of its impenetrability. Refusing to add port 53 (DNS) to the list of "white" ports does not guarantee that some attacker who can connect to the wireless network as a client will not try to circumvent such a rule, for example, by organizing their own Internet-accessible DNS server outside the school network that accepts requests not on standard port 53, but, for example, on port 443 (https). The firewall of the wireless network server will be powerless in this case. A simpler attack is also possible: an attacker has a record of the forbidden resource of interest in the dns cache of their device and connects to it without using the domain name server. It is even easier for an attacker to access an external resource using a URL containing the IP address of that resource. For the same reason, you can not exclude access to external proxy servers, for example, the firewall can not determine what is hidden behind, for example, 198.51.100.34:443. This can be the web interface of the mail server, the library site, or a proxy server that accepts connections on this port.



## The structure of the network filter

Based on the above and guided by the principle of necessity and sufficiency, we can characterize the structure of the network filter as follows:

- defined a set of ports that are considered secure,
- defined filtering tables for IPv4 and IPv6 protocols,
- each table defines a chain of rules for incoming, outgoing, and forwarded packets,
- the default packet drop policy is set for all IPv6 table chains,
- the default packet drop policy is defined for the incoming and forward chains of the IPv4 table, and the accept policy is set for the outgoing chain by default.
- packets related to already established connections are allowed to income and pass through,
- finally, enabling rules have been introduced for the traffic described above.

As a result, the script for atomic loading of rules took the form:

```
root@server:/# cat /etc/nftables.conf
#!/usr/sbin/nft -f
```

```
flush ruleset
```

```
define safe_ports = {http, https, ftp, ftp-data, ftps, ftps-data}
```

```
table ip filter {
  chain input {
    type filter hook input priority 0;
    policy drop;
    iiif lo accept
    ct state established,related accept
    iiif enp7s0 icmp type echo-request limit rate 10/second accept
    iiif wlp6s0 icmp type echo-request limit rate 100/second accept
    iiif enp7s0 ether saddr 90:2b:34:48:08:b5 ip saddr 192.168.0.2 tcp dport ssh ct
state new limit rate 1/second accept
    iiif wlp6s0 udp dport bootps limit rate 300/second accept
    iiif wlp6s0 udp dport domain limit rate 300/second accept
    iiif wlp6s0 tcp dport domain limit rate 300/second accept
    iiif wlp6s0 udp dport ntp limit rate 300/second accept
  }
  chain output {
    type filter hook output priority 0;
    policy accept;
  }
  chain forward {
    type filter hook forward priority 0;
    policy drop;
    ct state established,related accept
    ip saddr 192.168.4.0/24 ip daddr 192.168.7.0/24 accept
    ip saddr 192.168.4.0/24 ip daddr 192.168.0.4 tcp dport 8082 accept
    ip saddr 192.168.4.0/24 tcp dport $safe_ports accept
  }
}

table ip6 filter {
  chain input {
    type filter hook input priority 0;
    policy drop;
  }
  chain output {
    type filter hook output priority 0;
```

```

        policy drop;
    }
    chain forward {
        type filter hook forward priority 0;
        policy drop;
    }
}

```

## Starting and checking the network filter

Automatic loading of firewall rules is provided by the nftables package installer and does not require manual intervention. You can check the current set of rules using the command

```
root@debian:/# /usr/sbin/nft list ruleset
```

The output of this command, accurate to comments, pre-cleaning instructions, and substituting the list of allowed ports, must match the above scenario for loading firewall rules.

To make sure that the rules were loaded before starting the network interfaces, you can view the contents of the file **/var/log/daemon.log**, an excerpt from which, with some abbreviations, but preserving the order of the presented ones, looks like:

```

May 14 09:11:28 server systemd[1]: Starting nftables...
May 14 09:11:28 server systemd[1]: Starting Show Plymouth Boot Screen...
May 14 09:11:28 server systemd[1]: Started nftables.
...
May 14 09:11:28 server systemd[1]: Reached target Local File Systems.
May 14 09:11:28 server systemd[1]: Starting Raise network interfaces...
May 14 09:11:28 server systemd[1]: Reached target System Initialization.
May 14 09:11:33 server systemd[1]: Started Raise network interfaces.
...
May 14 09:11:33 server systemd[1]: Reached target Network.
...

```

You can also view the service status using the command:

```

root@server:/# systemctl status nftables
• nftables.service - nftables
   Loaded: loaded (/lib/systemd/system/nftables.service; enabled; vendor preset:
   enabled)
   Active: active (exited) since Thu 2020-05-14 09:11:28 MSK; 19min ago
     Docs: man:nft(8)
           http://wiki.nftables.org
   Process: 1633 ExecStart=/usr/sbin/nft -f /etc/nftables.conf (code=exited,
   status=0/SUCCESS)
  Main PID: 1633 (code=exited, status=0/SUCCESS)

мая 14 09:11:28 server systemd[1]: Starting nftables...
мая 14 09:11:28 server systemd[1]: Started nftables.

```

The firewall state check can be divided into a trivial check of the availability of allowed services to the client, including those located outside the wireless subnet, and a check by the Nmap network scanner of the availability of certain server ports from three different directions:

- from the system administrator's computer
- from any computer in the network core
- from a client computer on a wireless subnet

Testing is automated using a script /tmp/nmap.sh, whose contents are transparently guessed from its output, since the commands that are executed are displayed with the user@host prefix.

Scan protocol from the system administrator's computer:

```
root@admin:/tmp# chmod a+x nmap.sh
root@admin:/tmp# ./nmap.sh
Пинг-сканирование
user@host:/# nmap -sP 192.168.0.44
Starting Nmap 7.70 ( https://nmap.org ) at 2020-05-25 12:00 MSK
Nmap scan report for server.wifi.school34 (192.168.0.44)
Host is up (0.00033s latency).
MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))
Nmap done: 1 IP address (1 host up) scanned in 0.30 seconds
TCP SYN пинг
user@host:/# nmap -PS 192.168.0.44
Starting Nmap 7.70 ( https://nmap.org ) at 2020-05-25 12:00 MSK
Nmap scan report for server.wifi.school34 (192.168.0.44)
Host is up (0.00029s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Nmap done: 1 IP address (1 host up) scanned in 10.64 seconds
TCP ACK пинг
user@host:/# nmap -PA 192.168.0.44
Starting Nmap 7.70 ( https://nmap.org ) at 2020-05-25 12:00 MSK
Nmap scan report for server.wifi.school34 (192.168.0.44)
Host is up (0.00031s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Nmap done: 1 IP address (1 host up) scanned in 12.05 seconds
UDP пинг
user@host:/# nmap -PU 192.168.0.44
Starting Nmap 7.70 ( https://nmap.org ) at 2020-05-25 12:00 MSK
Nmap scan report for server.wifi.school34 (192.168.0.44)
Host is up (0.00032s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Nmap done: 1 IP address (1 host up) scanned in 12.11 seconds
Различные типы пинг-пакетов ICMP
user@host:/# nmap -PE 192.168.0.44
Starting Nmap 7.70 ( https://nmap.org ) at 2020-05-25 12:00 MSK
Nmap scan report for server.wifi.school34 (192.168.0.44)
Host is up (0.00031s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Nmap done: 1 IP address (1 host up) scanned in 12.08 seconds
user@host:/# nmap -PP 192.168.0.44
Starting Nmap 7.70 ( https://nmap.org ) at 2020-05-25 12:00 MSK
Nmap scan report for server.wifi.school34 (192.168.0.44)
Host is up (0.00027s latency).
```

Not shown: 999 filtered ports

PORT STATE SERVICE

22/tcp open ssh

MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Nmap done: 1 IP address (1 host up) scanned in 4.46 seconds

**user@host:/# nmap -PM 192.168.0.44**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:00 MSK

Nmap scan report for server.wifi.school34 (192.168.0.44)

Host is up (0.00029s latency).

Not shown: 999 filtered ports

PORT STATE SERVICE

22/tcp open ssh

MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Nmap done: 1 IP address (1 host up) scanned in 10.42 seconds

**Пинг с использование протокола IP**

**user@host:/# nmap -PO 192.168.0.44**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:01 MSK

Nmap scan report for server.wifi.school34 (192.168.0.44)

Host is up (0.00029s latency).

Not shown: 999 filtered ports

PORT STATE SERVICE

22/tcp open ssh

MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Nmap done: 1 IP address (1 host up) scanned in 4.48 seconds

**ARP пинг**

**user@host:/# nmap -PR 192.168.0.44**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:01 MSK

Nmap scan report for server.wifi.school34 (192.168.0.44)

Host is up (0.00031s latency).

Not shown: 999 filtered ports

PORT STATE SERVICE

22/tcp open ssh

MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Nmap done: 1 IP address (1 host up) scanned in 18.38 seconds

**TCP connect сканирование**

**user@host:/# nmap -sT 192.168.0.44**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:01 MSK

Nmap scan report for server.wifi.school34 (192.168.0.44)

Host is up (0.00036s latency).

Not shown: 999 filtered ports

PORT STATE SERVICE

22/tcp open ssh

MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Nmap done: 1 IP address (1 host up) scanned in 4.85 seconds

**UDP сканирование**

**user@host:/# nmap -sU 192.168.0.44**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:01 MSK

Nmap scan report for server.wifi.school34 (192.168.0.44)

Host is up (0.00027s latency).

All 1000 scanned ports on server.wifi.school34 (192.168.0.44) are open|filtered

MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Nmap done: 1 IP address (1 host up) scanned in 21.52 seconds

**TCP NULL сканирование**

**user@host:/# nmap -sN 192.168.0.44**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:01 MSK

Nmap scan report for server.wifi.school34 (192.168.0.44)

Host is up (0.00029s latency).  
All 1000 scanned ports on server.wifi.school34 (192.168.0.44) are open|filtered  
MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Nmap done: 1 IP address (1 host up) scanned in 21.46 seconds  
**TCP FIN сканирование**  
**user@host:/# nmap -sF 192.168.0.44**  
Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:02 MSK  
Nmap scan report for server.wifi.school34 (192.168.0.44)  
Host is up (0.00030s latency).  
All 1000 scanned ports on server.wifi.school34 (192.168.0.44) are open|filtered  
MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Nmap done: 1 IP address (1 host up) scanned in 21.47 seconds  
**TCP Xmas сканирование**  
**user@host:/# nmap -sX 192.168.0.44**  
Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:02 MSK  
Nmap scan report for server.wifi.school34 (192.168.0.44)  
Host is up (0.00026s latency).  
All 1000 scanned ports on server.wifi.school34 (192.168.0.44) are open|filtered  
MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Nmap done: 1 IP address (1 host up) scanned in 21.56 seconds  
**TCP ACK сканирование**  
**user@host:/# nmap -sA 192.168.0.44**  
Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:03 MSK  
Nmap scan report for server.wifi.school34 (192.168.0.44)  
Host is up (0.00029s latency).  
Not shown: 999 filtered ports  
PORT STATE SERVICE  
22/tcp unfiltered ssh  
MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Nmap done: 1 IP address (1 host up) scanned in 10.78 seconds  
**TCP Window сканирование**  
**user@host:/# nmap -sW 192.168.0.44**  
Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:03 MSK  
Nmap scan report for server.wifi.school34 (192.168.0.44)  
Host is up (0.00027s latency).  
Not shown: 999 filtered ports  
PORT STATE SERVICE  
22/tcp closed ssh  
MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Nmap done: 1 IP address (1 host up) scanned in 4.50 seconds  
**TCP сканирование Мэймона**  
**user@host:/# nmap -sM 192.168.0.44**  
Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:03 MSK  
Nmap scan report for server.wifi.school34 (192.168.0.44)  
Host is up (0.00026s latency).  
All 1000 scanned ports on server.wifi.school34 (192.168.0.44) are open|filtered  
MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Nmap done: 1 IP address (1 host up) scanned in 21.47 seconds  
**Сканирование протокола IP**  
**user@host:/# nmap -sO 192.168.0.44**  
Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:03 MSK  
Nmap scan report for server.wifi.school34 (192.168.0.44)  
Host is up (0.00026s latency).  
Not shown: 255 open|filtered protocols  
PROTOCOL STATE SERVICE  
1 open icmp

MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Nmap done: 1 IP address (1 host up) scanned in 3.48 seconds

**Определение версий ОС и служб в сочетании с TCP SYN сканирование**

**user@host:/# nmap -O -sV -sS 192.168.0.44**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:03 MSK

Nmap scan report for server.wifi.school34 (192.168.0.44)

Host is up (0.00031s latency).

Not shown: 999 filtered ports

PORT STATE SERVICE VERSION

22/tcp open ssh OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)

MAC Address: B8:70:F4:29:E5:83 (Compal Information (kunshan))

Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port

Device type: general purpose

Running: Linux 2.6.X

OS CPE: cpe:/o:linux:linux\_kernel:2.6

OS details: Linux 2.6.18 - 2.6.22

Network Distance: 1 hop

Service Info: OS: Linux; CPE: cpe:/o:linux:linux\_kernel

OS and Service detection performed. Please report any incorrect results at

<https://nmap.org/submit/> .

Nmap done: 1 IP address (1 host up) scanned in 13.10 seconds

Scan protocol from a computer in the network core (namely, from a virtualization server):

**Пинг-сканирование**

**user@host:/# nmap -sP 192.168.0.44**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK

Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn

Nmap done: 1 IP address (0 hosts up) scanned in 0.48 seconds

**TCP SYN пинг**

**user@host:/# nmap -PS 192.168.0.44**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK

Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn

Nmap done: 1 IP address (0 hosts up) scanned in 0.52 seconds

**TCP ACK пинг**

**user@host:/# nmap -PA 192.168.0.44**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK

Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn

Nmap done: 1 IP address (0 hosts up) scanned in 0.51 seconds

**UDP пинг**

**user@host:/# nmap -PU 192.168.0.44**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK

Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn

Nmap done: 1 IP address (0 hosts up) scanned in 0.51 seconds

**Различные типы пинг-пакетов ICMP**

**user@host:/# nmap -PE 192.168.0.44**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK

Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn

Nmap done: 1 IP address (0 hosts up) scanned in 0.51 seconds

**user@host:/# nmap -PP 192.168.0.44**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK

Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn

Nmap done: 1 IP address (0 hosts up) scanned in 0.51 seconds

**user@host:/# nmap -PM 192.168.0.44**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK

Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn

Nmap done: 1 IP address (0 hosts up) scanned in 0.52 seconds

**Пинг с использование протокола IP**

**user@host:/# nmap -PO 192.168.0.44**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK  
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn  
Nmap done: 1 IP address (0 hosts up) scanned in 0.51 seconds

**ARP пинг**  
**user@host:/# nmap -PR 192.168.0.44**  
Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK  
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn  
Nmap done: 1 IP address (0 hosts up) scanned in 0.52 seconds

**TCP connect сканирование**  
**user@host:/# nmap -sT 192.168.0.44**  
Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK  
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn  
Nmap done: 1 IP address (0 hosts up) scanned in 0.50 seconds

**UDP сканирование**  
**user@host:/# nmap -sU 192.168.0.44**  
Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK  
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn  
Nmap done: 1 IP address (0 hosts up) scanned in 0.50 seconds

**TCP NULL сканирование**  
**user@host:/# nmap -sN 192.168.0.44**  
Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK  
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn  
Nmap done: 1 IP address (0 hosts up) scanned in 0.51 seconds

**TCP FIN сканирование**  
**user@host:/# nmap -sF 192.168.0.44**  
Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK  
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn  
Nmap done: 1 IP address (0 hosts up) scanned in 0.51 seconds

**TCP Xmas сканирование**  
**user@host:/# nmap -sX 192.168.0.44**  
Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK  
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn  
Nmap done: 1 IP address (0 hosts up) scanned in 0.51 seconds

**TCP ACK сканирование**  
**user@host:/# nmap -sA 192.168.0.44**  
Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK  
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn  
Nmap done: 1 IP address (0 hosts up) scanned in 0.50 seconds

**TCP Window сканирование**  
**user@host:/# nmap -sW 192.168.0.44**  
Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK  
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn  
Nmap done: 1 IP address (0 hosts up) scanned in 0.51 seconds

**TCP сканирование Мэймона**  
**user@host:/# nmap -sM 192.168.0.44**  
Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK  
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn  
Nmap done: 1 IP address (0 hosts up) scanned in 0.50 seconds

**Сканирование протокола IP**  
**user@host:/# nmap -sO 192.168.0.44**  
Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK  
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn  
Nmap done: 1 IP address (0 hosts up) scanned in 0.48 seconds

**Определение версий ОС и служб в сочетании с TCP SYN сканирование**  
**user@host:/# nmap -O -sV -sS 192.168.0.44**  
Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 12:28 MSK  
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn  
Nmap done: 1 IP address (0 hosts up) scanned in 0.99 seconds

The additional scan recommended by the nmap output gave the same result:

**user@host:/# nmap -Pn 192.168.0.44**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-25 13:41 MSK  
Nmap done: 1 IP address (0 hosts up) scanned in 0.52 seconds

Scan protocol from the client computer:

**Пинг-сканирование**

**user@host:/# nmap -sP 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 13:58 MSK  
Nmap scan report for 192.168.4.1  
Host is up (0.0057s latency).  
MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)  
Nmap done: 1 IP address (1 host up) scanned in 0.28 seconds

**TCP SYN пинг**

**user@host:/# nmap -PS 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 13:58 MSK  
Nmap scan report for 192.168.4.1  
Host is up (0.0013s latency).  
Not shown: 999 filtered ports  
PORT STATE SERVICE  
53/tcp open domain  
MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 10.34 seconds

**TCP ACK пинг**

**user@host:/# nmap -PA 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 13:58 MSK  
Nmap scan report for 192.168.4.1  
Host is up (0.0013s latency).  
Not shown: 999 filtered ports  
PORT STATE SERVICE  
53/tcp open domain  
MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 7.85 seconds

**UDP пинг**

**user@host:/# nmap -PU 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 13:58 MSK  
Nmap scan report for 192.168.4.1  
Host is up (0.0013s latency).  
Not shown: 999 filtered ports  
PORT STATE SERVICE  
53/tcp open domain  
MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 13.33 seconds

**Различные типы пинг-пакетов ICMP**

**user@host:/# nmap -PE 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 13:58 MSK  
Nmap scan report for 192.168.4.1  
Host is up (0.0012s latency).  
Not shown: 999 filtered ports  
PORT STATE SERVICE  
53/tcp open domain  
MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 6.69 seconds

**user@host:/# nmap -PP 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 13:58 MSK  
Nmap scan report for 192.168.4.1  
Host is up (0.0019s latency).  
Not shown: 999 filtered ports



PORT STATE SERVICE  
53/tcp open domain  
MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 12.20 seconds

**user@host:/# nmap -PM 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 13:59 MSK

Nmap scan report for 192.168.4.1

Host is up (0.0013s latency).

Not shown: 999 filtered ports

PORT STATE SERVICE

53/tcp open domain

MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 7.90 seconds

**Пинг с использование протокола IP**

**user@host:/# nmap -PO 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 13:59 MSK

Nmap scan report for 192.168.4.1

Host is up (0.0013s latency).

Not shown: 999 filtered ports

PORT STATE SERVICE

53/tcp open domain

MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 12.39 seconds

**ARP пинг**

**user@host:/# nmap -PR 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 13:59 MSK

Nmap scan report for 192.168.4.1

Host is up (0.0013s latency).

Not shown: 999 filtered ports

PORT STATE SERVICE

53/tcp open domain

MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 12.81 seconds

**TCP connect сканирование**

**user@host:/# nmap -sT 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 13:59 MSK

Nmap scan report for 192.168.4.1

Host is up (0.0024s latency).

Not shown: 999 filtered ports

PORT STATE SERVICE

53/tcp open domain

MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 4.83 seconds

**UDP сканирование**

**user@host:/# nmap -sU 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 13:59 MSK

Nmap scan report for 192.168.4.1

Host is up (0.0011s latency).

Not shown: 999 open|filtered ports

PORT STATE SERVICE

123/udp open ntp

MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 9.14 seconds

**TCP NULL сканирование**

**user@host:/# nmap -sN 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 13:59 MSK

Nmap scan report for 192.168.4.1  
Host is up (0.0011s latency).  
All 1000 scanned ports on 192.168.4.1 are open|filtered  
MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 21.45 seconds

#### **TCP FIN сканирование**

**user@host:/# nmap -sF 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 14:00 MSK

Nmap scan report for 192.168.4.1

Host is up (0.0011s latency).

All 1000 scanned ports on 192.168.4.1 are open|filtered

MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 21.47 seconds

#### **TCP Xmas сканирование**

**user@host:/# nmap -sX 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 14:00 MSK

Nmap scan report for 192.168.4.1

Host is up (0.0011s latency).

All 1000 scanned ports on 192.168.4.1 are open|filtered

MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 21.46 seconds

#### **TCP ACK сканирование**

**user@host:/# nmap -sA 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 14:01 MSK

Nmap scan report for 192.168.4.1

Host is up (0.0021s latency).

Not shown: 999 filtered ports

PORT	STATE	SERVICE
------	-------	---------

53/tcp	unfiltered	domain
--------	------------	--------

MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 12.22 seconds

#### **TCP Window сканирование**

**user@host:/# nmap -sW 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 14:01 MSK

Nmap scan report for 192.168.4.1

Host is up (0.0016s latency).

Not shown: 999 filtered ports

PORT	STATE	SERVICE
------	-------	---------

53/tcp	closed	domain
--------	--------	--------

MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 18.08 seconds

#### **TCP сканирование Мэймона**

**user@host:/# nmap -sM 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 14:01 MSK

Nmap scan report for 192.168.4.1

Host is up (0.0013s latency).

Not shown: 999 open|filtered ports

PORT	STATE	SERVICE
------	-------	---------

53/tcp	closed	domain
--------	--------	--------

MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 10.78 seconds

#### **Сканирование протокола IP**

**user@host:/# nmap -sO 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 14:01 MSK

Nmap scan report for 192.168.4.1

Host is up (0.0015s latency).

Not shown: 255 open|filtered protocols

PROTOCOL STATE SERVICE

1 open icmp

MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 4.58 seconds

**Определение версий ОС и служб в сочетании с TCP SYN сканирование**

**user@host:/# nmap -O -sV -sS 192.168.4.1**

Starting Nmap 7.70 ( <https://nmap.org> ) at 2020-05-28 14:01 MSK

Nmap scan report for 192.168.4.1

Host is up (0.0015s latency).

Not shown: 999 filtered ports

PORT STATE SERVICE VERSION

53/tcp open domain dnsmasq 2.80

MAC Address: D0:DF:9A:6A:89:18 (Liteon Technology)

Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port

Device type: general purpose

Running: Linux 3.X|4.X

OS CPE: cpe:/o:linux:linux\_kernel:3 cpe:/o:linux:linux\_kernel:4

OS details: Linux 3.10 - 4.11, Linux 3.2 - 4.9

Network Distance: 1 hop

OS and Service detection performed. Please report any incorrect results at <https://nmap.org/submit/> .

Nmap done: 1 IP address (1 host up) scanned in 20.59 seconds

Analyzing the network scanner protocols, you can draw the following conclusions:

When scanning the system from the system administrator's computer, only the ssh port was available, which was expected. In addition, it was possible to check the availability of the system using the ICMP Protocol and get its hardware address, which also does not contradict the task statement and expected results. When determining software versions, the ssh service not only allowed you to determine its own version, but also reported the operating system version. However, this should not be considered a drawback, since it was not possible to get this information from the network core when checking the system from a third-party computer.

When the wireless network client scanned the system, only ports 123/udp (network time service) and 53/tcp (domain name service) were detected. When determining the software versions, it was found out that DNS server services are provided by dnsmasq version 2.80, running on Linux OS with kernel version 3. X-4. X. Such results should also be considered fully compliant with the requirements.

In addition to protecting the wireless network server itself, the firewall must also improve the security of the network's clients. Thus, an attempt to find a route from the virtualization server to one of the client devices with the address 192.168.4.29 failed, as did an attempt to detect this client by the ping utility:

**root@virtserver:/# traceroute 192.168.4.29**

traceroute to 192.168.4.29 (192.168.4.29), 30 hops max, 60 byte packets

1 192.168.0.1 (192.168.0.1) 0.547 ms 0.501 ms 0.468 ms

2 server.wifi.school34 (192.168.0.44) 0.415 ms 0.384 ms 0.359 ms

3 \* \* \*

...

30 \* \* \*

**root@virtserver:/# ping 192.168.4.29**

PING 192.168.4.29 (192.168.4.29) 56(84) bytes of data.

From 192.168.0.1: icmp\_seq=2 Redirect Host(New nexthop: 192.168.0.44)

```
...
^C
--- 192.168.4.29 ping statistics ---
21 packets transmitted, 0 received, 100% packet loss, time 482ms
```

*Note:* here the name of the virtualization server (server, server.service.school34) in the command prompt is changed to avoid confusion with the name of the wireless network server (server, server.wifi.school34) to virtserver.

However, the client has full access to the virtual servers located on this virtualization server.

At the same time, the client does not have access to devices in other subnets, and its communication with other devices in the wireless network is not possible if the wireless network server is disconnected from it (for example, using the power management tools of the network card).

## Completing the installation

When the installation is complete, it makes sense to update the system, clear the local package cache, and delete packages that the system no longer needs. You can do this with commands:

```
root@server:/# apt-get upgrade
root@server:/# apt-get autoremove
root@server:/# apt-get clean
```

## Using the server

The scenarios, commands, and techniques described below are intended to be used during the operation of the system to monitor its state or change operating modes. The following description can be taken as a quick guide (how to).

## Login procedure

Due to the ssh and network filter settings described above, the following remote login procedure is provided: log in from the system administrator's computer as guest, and then, if necessary, upgrade privileges to root locally. These actions are shown in the following fragment of the terminal session:

```
administrator@admin:~$ ssh guest@server.wifi.school34
guest@server.wifi.school34's password:
Last login: Tue Mar  3 14:26:42 2020 from 192.168.0.2
```

The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/\*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

```
guest@server:~$ su
Password:
root@server:/home/guest#
```

To use programs with a graphical user interface, use the -X key in the ssh connection creation command:

```
administrator@admin:~$ ssh -X guest@server.wifi.school34
```

## Hardware health monitoring

To get information from hardware sensors, just run the sensors command on behalf of any user, for example:

```
guest@server:/# sensors
coretemp-isa-0000
Adapter: ISA adapter
Core 0:      +40.0°C (high = +80.0°C, crit = +90.0°C)
Core 2:      +43.0°C (high = +80.0°C, crit = +90.0°C)
acpitz-acpi-0
Adapter: ACPI interface
temp1:       +42.0°C (crit = +127.0°C)
nouveau-pci-0100
Adapter: PCI adapter
GPU core:    +0.85 V (min = +0.80 V, max = +1.00 V)
temp1:       +50.0°C (high = +95.0°C, hyst = +3.0°C)
               (crit = +105.0°C, hyst = +5.0°C)
               (emerg = +135.0°C, hyst = +5.0°C)
```

To get information from the built in hard disk self diagnostics hardware you can use the following superuser commands:

Search for all S. M. A. R. T-compatible devices:

```
root@server:/# smartctl --scan
/dev/sda -d scsi # /dev/sda, SCSI device
```

Obtaining information about the device:

```
root@server:/# smartctl /dev/sda -i
smartctl 6.6 2017-11-05 r4594 [x86_64-linux-4.19.0-5-amd64] (local build)
Copyright (C) 2002-17, Bruce Allen, Christian Franke, www.smartmontools.org
```

```
=== START OF INFORMATION SECTION ===
Model Family:      Seagate Momentus 5400.6
Device Model:      ST9320325AS
Serial Number:     6VDCABKR
LU WWN Device Id:  5 000c50 03dc5905c
Firmware Version:  0011LVM1
User Capacity:     320 072 933 376 bytes [320 GB]
Sector Size:       512 bytes logical/physical
Rotation Rate:     5400 rpm
Device is:         In smartctl database [for details use: -P show]
ATA Version is:    ATA8-ACS T13/1699-D revision 4
SATA Version is:   SATA 2.6, 3.0 Gb/s (current: 3.0 Gb/s)
Local Time is:     Thu Mar 19 11:31:16 2020 MSK
SMART support is:  Available - device has SMART capability.
SMART support is:  Enabled
```

Getting detailed information about the device related to S. M. A. R. T (the command output is omitted for brevity):

```
root@server:/# smartctl /dev/sda -a
```

Getting even more detailed information about the device related to S. M. A. R. T and beyond (even more extensive output of the command is also omitted for brevity):

```
root@server:/# smartctl /dev/sda -x
```

In addition, you can use the graphical user interface for these and other commands – GSmartControl. However, running this utility also requires knowing the superuser password.

## Changing the external domain name server

When a wireless network server is running, it may often be necessary to change the external DNS server. In the main usage mode, it uses YandexDNS services with minimal filtering of resources by their domain names, but it may often be necessary to strengthen the filtering mode. To do this, you need to switch to a more "strict" SkyDNS. The switching process consists of three steps:

- changing the external server address in the main dnsmasq configuration file
- restarting the dnsmasq service
- verifying the success of restarting the service

To simplify this process, the system has the **/usr/local/sbin/dnsmode** script :

```
#!/bin/bash

if [ $1 = "free" ]
then
    sed s/"193\.58\.251\.251"/"77\.88\.8\.7"/ -i /etc/dnsmasq.conf
    systemctl restart dnsmasq
elif [ $1 = "safe" ]
then
    sed s/"77\.88\.8\.7"/"193\.58\.251\.251"/ -i /etc/dnsmasq.conf
    systemctl restart dnsmasq
fi
systemctl status --no-pager dnsmasq
exit 0
```

This script should be used with a single command-line parameter that takes the values «free» or «safe». If you specify any other values (including an empty one), the script only outputs the current state of the service. If the value is «free», it switches to YandexDNS, and if the value is «safe», it switches to SkyDNS.

The following mixed (in the sense of executing commands both on the server and on the client) fragment of the terminal session demonstrates the behavior of the script:

```
root@server:/# dnsmode free
• dnsmasq.service - dnsmasq - A lightweight DHCP and caching DNS server
  Loaded: loaded (/lib/systemd/system/dnsmasq.service; enabled; vendor preset:
enabled)
  Drop-In: /etc/systemd/system/dnsmasq.service.d
    └─override.conf
  Active: active (running) since Tue 2020-03-03 15:06:04 MSK; 9ms ago
...
map 03 15:06:04 server dnsmasq[3752]: using nameserver 77.88.8.7#53(via enp7s0)

xubuntu@xubuntu:~$ host vk.com
vk.com has address 87.240.139.194
vk.com has address 87.240.190.67
vk.com has address 87.240.190.72
vk.com has address 93.186.225.208
vk.com has address 87.240.137.158
vk.com has address 87.240.190.78
```

```
vk.com mail is handled by 0 mx.vk.com.  
vk.com mail is handled by 20 mx2.vk.com.
```

```
root@server:/# dnsmode safe
```

```
• dnsmasq.service - dnsmasq - A lightweight DHCP and caching DNS server  
Loaded: loaded (/lib/systemd/system/dnsmasq.service; enabled; vendor preset:  
enabled)
```

```
Drop-In: /etc/systemd/system/dnsmasq.service.d  
└─override.conf
```

```
Active: active (running) since Tue 2020-03-03 15:06:51 MSK; 8ms ago
```

```
...
```

```
map 03 15:06:51 server dnsmasq[3782]: using nameserver 193.58.251.251#53(via enp7s0)
```

```
xubuntu@xubuntu:~$ host vk.com
```

```
vk.com has address 193.58.251.1
```

So first the DNS service is switched to free mode, then the client performs a dns query about the social network domain, which receives a full response. Then the server is switched to filtering mode and the same request is executed on the client. Now the client gets the address of the SkyDNS blocking page instead of real addresses.

It should be noted, first, that the script output explicitly specifies the external domain name server being used.

Secondly, it should be clarified that connections already established by the client are not broken during this switch, only the possibility of new connections with unreliable resources is blocked.

## Getting the current list of client devices

When connecting a client and giving a network address, information about this client is written in the file `/var/lib/misc/dnsmasq.leases`. The format of this file is as follows: each client is assigned a single line that sequentially lists the time until the end of the address lease term, the client's hardware address, the IPv4 address assigned to it (the system in question does not work with IPv6), the client host name, its own or leased along with the address, and the DHCP client ID. At some point in the system's operation, this file looked like:

```
root@server:/# cat /var/lib/misc/dnsmasq.leases
```

```
1583282460 cc:61:e5:fa:e1:cd 192.168.4.29 android-8a69d96bb881b05c *
```

```
1583279954 d0:df:9a:6a:4c:b4 192.168.4.16 debian *
```

This corresponds to an android smartphone and laptop with the hostname «debian».

Viewing this file, accordingly, allows you to identify clients who have received the address, but have not yet rejected it. However, it is not guaranteed that the client specified in this file is actually online at the moment. The client may be out of the wireless network coverage area and thus lose contact with it. However, in this case, the connection to the network was not closed correctly, and the client can return to it without re-passing the procedure for obtaining the address, of course, if the time for renting the address did not expire during his absence. On the other hand, the client device may permanently leave the network without correctly closing the connection, and information about it will still be contained in this file. Therefore, this file is not a guaranteed reliable source of information about clients currently on the network. In addition to it, you can use the command

```
root@server:/# ip neighbour show dev wlp6s0
```

The output of this command displays the hardware addresses of network devices known to the server (arp cache). Information about these devices is updated much more often than data about rented addresses, so this information can help to identify customers who are actually on the network or have left it relatively recently.

Another means of checking the presence of a client in the network is the ping command, but it is not guaranteed that a client device, for example, protected by a firewall, will respond to its request.

Earlier in the description of configuring the dnsmasq service, we also provided a method for sending the SIGUSR1 signal to the dnsmasq process.

Finally, using a network scanner (for example, nmap) will allow you to get a response with an even greater degree of confidence, but, again, without guarantees, because the client device is also able to counteract this.

However, setting aside situations that are too unlikely within given the network's operating conditions, it can be noted that the use of the first three tools in the vast majority of cases is sufficient to identify the current active wireless network clients.

## **Notes on connecting clients**

Finally, you should make a note about connecting client devices: only the client device itself determines the set of services that it is ready to use. The server can only offer these services. For example, the windows device dhcp client does not use the hostname received from the server. Other devices may well ignore both the domain name and the proposed network time and domain name servers. In most cases, this does not pose any difficulties for the network itself. The only exceptions are attempts to bypass filtering by domain names or attempts to use third-party proxy servers, which, however, is opposed by the firewall.

Also, the server functionality does not include the ability to provide customers with information about the services of other servers in the local network. Thus, shortcuts on desktops, bookmarks in Internet browsers, and so on are left to the discretion of the customers themselves.