

Debian 10 virtual repository



Contents

Problem statement.....	2
Preliminary observation.....	2
Creating a virtual machine.....	2
System installation and disk layout.....	3
Basic configuration of the virtual server.....	3
Disabling IPv6.....	4
Disabling Magic SysRq.....	4
Creating users.....	5
Configuring access lists.....	6
Network configuration.....	7
Disabling unnecessary services.....	8
Using a local mirror of the repository during setup.....	9
Installing auxiliary software.....	9
Installing and configuring ssh.....	10
Configuration.....	10
Fail protection (auto-resume).....	11
Verification of performance and stability.....	11
Installing and configuring lighttpd.....	13
Configuration.....	13
Fail protection (auto-resume).....	15
Verification of performance and stability.....	15
Creating a personal repository.....	16
Repository content protection.....	18
About additional services.....	19
Database server.....	19
Password protection (fail2ban).....	19
Log rotation (logrotate).....	20
Network time service (ntp).....	20
Mail service.....	20
Installing and configuring the network filter (nftables).....	21
Open port.....	21
Network environment.....	22
The structure of the network filter.....	22
Starting and checking the network filter.....	23
Completing the installation.....	28
Using a virtual server.....	28
How to change the virtualization server.....	28
Login procedure.....	29
How to synchronize with official repositories.....	29

How to manage a personal repository.....	30
Connecting client to the repositories.....	31

Problem statement

It is needed to create a virtual server for the QEmu/KVM hypervisor that implements the following repositories of the Debian 10 distribution for the amd64 architecture:

- mirror of the official repository
- mirror repository with security updates
- repository with own packages

Repositories must be accessed over the http protocol. Reasonable measures should be taken to protect the server when the server receives network settings over the dhcp protocol. Managing the server should be done via ssh connection. The server itself must also run Debian 10 (amd64).

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.

Since the virtual server in question will be used in multiple virtual networks, the abstract user, localhost, localdomain, and so on will be used instead of specific domain names and user accounts.

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.

Creating a virtual machine

The virtual machine was created using Virtual Machine Manager (virt-manager). When selecting virtual hardware settings, most of the parameters have retained their default values, note only the following:

A 200 GB qcow2 virtual hard disk has been created for the VM and was connected to the VM via the VirtIO bus.

The server is allocated one single-core processor that copies the configuration of the host system's processor core.

The server allocated 1 GB of RAM.

After installing the system, the virtual hard disk is left as the only bootable device.

The server network card is connected to a virtual network of servers that is open, routable, and does not use the hypervisor's built-in dhcp server.

System installation and disk layout

The system was installed without graphical shell and with a minimal set of software (basic system, standard utilities, ssh service).

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

```
root@debian:/# /usr/sbin/fdisk -l /dev/vda
Disk /dev/vda: 200 GiB, 214748364800 bytes, 419430400 sectors
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: 0x9f5bc31c

Device      Boot      Start          End      Sectors      Size Id Type
/dev/vda1   *              2048     19531775     19529728      9,3G 83 Linux
/dev/vda2                19531776     23437311      3905536       1,9G 83 Linux
/dev/vda3                23437312     25391103      1953792       954M 82 Linux swap / Solaris
/dev/vda4                25391104     419428351    394037248    187,9G 83 Linux
```

```
root@debian:/# mount -l
/dev/vda1 on / type ext4 (rw,relatime,errors=remount-ro)
/dev/vda4 on /var/www type ext4 (rw,nosuid,nodev,noexec,relatime)
/dev/vda2 on /tmp type ext4 (rw,nosuid,nodev,relatime)
```

```
root@debian:/# df -h
Файловая система  Размер  Использовано  Дост  Использовано%  Смонтировано в
/dev/vda1          9,2G      1,2G      7,6G             13% /
/dev/vda4         184G       70G     105G             40% /var/www
/dev/vda2          1,9G       5,7M      1,7G              1% /tmp
```

This disk layout scheme was chosen for the following reasons:

The DOS table is the simplest and most minimalistic, which is optimal for a virtual server. About 1 GB is allocated for the swap partition, and the rest of the disk space is divided into three partitions, so the repository directory and the temporary file directory are located in separate file systems. This is done as part of basic server protection measures. Most of the disk is reserved for storing repositories, while the rest of the system components are allocated a minimum of sufficient (with some margin) disk space. Keep in mind that the system logs are also located in the root file system.

Basic configuration of the virtual server

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

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 it doesn't contradict the following lines in it:

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

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

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

and in the output of the command

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

make sure 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: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 52:54:00:79:69:b1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.7.2/24 brd 192.168.7.255 scope global enp1s0
        valid_lft forever preferred_lft forever
```

Disabling Magic SysRq

Disabling Magic SysRq may seem redundant for the VM, or even harmful: for example, if the system crashes, it will not be possible to crash it or dump the contents of buffers to disk. However, keep in mind that the server is virtual, and the data stored on it are easily recoverable. At the same time, there is a very small chance that an attacker will be able to imitate such a key combination in one way or another. This may cause the virtual server to stop and client service to be denied. 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 from it

```
kernel.sysrq=0
```

Creating 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, you should create a regular user, on whose behalf the remote login will be performed. After opening a session in the system under such an account, you can, if necessary, increase your privileges using `su`. From the point of view of security, this mechanism is an additional barrier of protection: the attacker must guess not only the superuser password, but also the username and password of a normal user.

You can create such user (let's say its username is `maintainer`) both at the system installation stage and after, for example, using the command

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

In addition to creating a user, you need to pay attention to several aspects of its operation in the system.

First, by default, the newly created home directory of this user is readable by everyone in the system. Given that this folder will store the private part of the repository signature key, even if it is protected by a passphrase, you should change the access rights to it a little with the command

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

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 for him). For all others, access will be closed.

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 a 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 easiest (and most optimal on a virtual server with one user and one implemented service) to use the following approach: do not use `sudo` at all, forcing the user to increase their rights in the system to enter (and, accordingly, know) the superuser password.

Thus, to prevent the user from using `sudo`, it is enough to exclude him/her from the `wheel` group, if he/she is a member of it. You can do this with the `usermod` command, leaving the `maintainer` user only in his/her (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@debian:/home/maintainer# /usr/sbin/usermod -G maintainer maintainer
root@debian:/home/maintainer# su maintainer
maintainer@debian:~$ groups
maintainer
maintainer@debian:~$ exit
exit
root@debian:/home/maintainer#
```

In conclusion, it should be noted that this account can be used not only to provide login protection, but also to store auxiliary information: notes, own packages that have not yet been added to the personal repository, and so on.

Configuring access 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, but `lighttpd` does not:

```
root@debian:/# ldd /usr/sbin/lighttpd | grep libwrap
root@debian:/# ldd /sbin/sshd | grep libwrap
libwrap.so.0 => /lib/x86_64-linux-gnu/libwrap.so.0 (0x00007f8ea045c000)
```

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

```
root@debian:/# 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, given 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 immune from developer errors and vulnerabilities. The use of this mechanism may be an additional line of defense in case of compromise of the network filter.

In this system, it does not make sense to install `tcpd`, since `ssh` uses this mechanism independently, and there are no restrictions on the connection source for `lighttpd`.

Knowing that you can only connect via `ssh` from a subnet server with the address `192.168.7.1`, and from anywhere via `http`, you should change the `/etc/hosts.allow` file to the following format (comments are omitted for brevity):

```
lighttpd : ALL
sshd : 192.168.7.1
```

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. Of course, 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, the firewall of the virtualization server).

Network configuration

In this section, for brevity and clarity, the virtual repository server will be called the guest, and the virtualization server on which it is located and running will be called the host.

The guest receives network settings via the dhcp protocol. However, from the point of view of security and taking into account its location in the network, some measures should be taken. So, if we assume that a certain device (computer, virtual or physical server) in the same subnet has fallen under the full control of an attacker, then the guest in question must be able to resist attacks without relying solely on the security systems of the host on which it is located. The description of the network filter will be given below. it should also be noted here that the host's hardware address (lladdr, mac) must be known to the guest in all circumstances and must be recorded.

To achieve these goals, it is enough to bring the configuration file **/etc/network/interfaces** to the form on the guest in question (some comments are abbreviated):

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

```
auto lo
iface lo inet loopback
```

```
auto enp1s0
iface enp1s0 inet dhcp
    pre-up ip neigh add 192.168.7.1 dev enp1s0 lladdr 52:54:00:2b:1e:a0 nud permanent
```

and restart the network subsystem

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

Then you can verify that the settings you received are correct (of course, if the dhcp server is active and available) by running the following commands:

```
root@debian:/# 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: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
```

```
    link/ether 52:54:00:79:69:b1 brd ff:ff:ff:ff:ff:ff
```

```
    inet 192.168.7.2/24 brd 192.168.7.255 scope global enp1s0
```

```
        valid_lft forever preferred_lft forever
```

```
root@debian:/# ip neighbour show
```

```
192.168.7.1 dev enp1s0 lladdr 52:54:00:2b:1e:a0 PERMANENT
```

```
root@debian:/# ip route show
```

```
default via 192.168.7.1 dev enp1s0
```

```
192.168.7.0/24 dev enp1s0 proto kernel scope link src 192.168.7.2
```

```
root@debian:/# hostname -f
```

```
debian.service.localdomain
```

```
root@debian:/# dnsdomainname
```

```
service.localdomain
```

Here (in order of commands) it is shown that

- guest received an IPv4 address and network mask (192.168.7.2/24),
- guest permanently matches the host's hardware and network addresses (192.168.7.1 and 52: 54:00:2b:1e:a 0)
- guest has a configured routing table
- guest received the full domain name via dhcp (debian in service.localdomain)

A number of comments should be made about assigning a full domain name:

First, the top-level domain (here localdomain) should not cause confusion: if the dns server of the local network is configured correctly, the use of such domains is possible and the mention of them will not go beyond it and will not cause inconvenience to external dns servers.

Second, in order for the guest to accept and use the domain name from the dhcp server, the guest should not be assigned the host and domain names manually, i.e. the **/etc/hostname** file must contain only the following:

```
localhost
```

just like the **/etc/hosts** file, it must contain the following information:

```
127.0.0.1      localhost
127.0.1.1      localhost
```

Other information in this case is not needed in this file: neither information about other network hosts, nor lines filled in by the system during installation concerning IPv6 settings, since support for this protocol was disabled earlier.

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@debian:/# dpkg -l | grep rpcbind
root@debian:/# which rsh
root@debian:/# 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@debian:/# which rsh
/usr/bin/rsh
root@debian:/home/maintainer# ss -Htlp
LISTEN  0    128      0.0.0.0:http      0.0.0.0:*    users:(("lighttpd",pid=362,fd=4))
LISTEN  0    128      0.0.0.0:ssh       0.0.0.0:*    users:(("sshd",pid=388,fd=3))
root@debian:/home/maintainer# ss -Hulp
UNCONN  0    0        0.0.0.0:bootpc    0.0.0.0:*    users:(("dhclient",pid=351,fd=7))
root@debian:/# dpkg -l | grep rsh
root@debian:/# ls -l /usr/bin | grep rsh
lrwxrwxrwx 1 root root          21 июл 17 18:40 rsh -> /etc/alternatives/rsh
root@debian:/# ls -l /etc/alternatives | grep rsh
lrwxrwxrwx 1 root root    12 июл 17 18:40 rsh -> /usr/bin/ssh
lrwxrwxrwx 1 root root    28 июл 17 18:40 rsh.1.gz -> /usr/share/man/man1/ssh.1.gz
```


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 and third commands indicate that the rsh daemon does not listen to any tcp or udp ports. The following command informs you that there are no packages installed on the system that mention rsh in their names. 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.

It should be noted that during minimalistic installation none of these services have been installed initially.

Using a local mirror of the repository during setup

Since the server in question was created as a mirror of the main Debian 10 repository, as well as taking into account that the corresponding repository was previously mirrored on removable media, it is possible to use this local mirror to configure the server itself without using the Internet channel, which, in general, could not be considered secure before the server configuration completes.

To do this it is enough to mount a removable media and copy the repository mirrors (buster and security) to the destination directories:

```
root@debian:/# mkdir /tmp/disk && mount /dev/sda1/ /tmp/disk && cd /tmp/disk
root@debian:/# cp -r /tmp/disk/buster /var/www
root@debian:/# cp -r /tmp/disk/security /var/www
```

However, after this, all files and directories are available for writing, reading, and execution to everyone in the system. This is not a problem while configuring the server, but you should change this before commissioning the server. These changes are justified and described in detail below.

Then, to use local mirrors, and later a personal repository, it is enough to add the following lines to the **/etc/apt/sources.list** file:

```
deb file:/var/www/buster buster main contrib non-free
deb file:/var/www/security buster/updates main contrib
deb file:/var/www/ppa buster main contrib
```

Given the role of the repository in the local network and the nature of its use, you can leave these three lines as the only contents of this file.

It should also be noted that in this mode of access to the repository, the server itself does not need the services of its own web server and can use the repository immediately after entering these lines in the sources file.

Installing auxiliary software

The Midnight Commander file manager (mc package), the tree utility, and the synchronization tool with official debmirror repositories can be very useful when managing the repository server. To install them, just run the following commands:

```
root@debian:/# apt-get update
root@debian:/# apt-get install mc tree debmirror
```

Installing and configuring ssh

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

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

On this server, ssh is used as the only remote login and administration tool. Given the server's location on the network, ssh accepts connections on the same network interface as the repository web server. Therefore, 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, in fact, the configuration of the ssh service itself.

The configuration of network shielding is described below, and the basic configuration of the server is described above when it was said about 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
Protocol 2
PermitRootLogin no
AllowUsers maintainer
PasswordAuthentication yes
PubkeyAuthentication no
KerberosAuthentication no
HostbasedAuthentication no
IgnoreRhosts yes
PermitEmptyPasswords no
X11Forwarding no
AddressFamily inet
```

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 – uses ssh version 2.

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

AllowUsers maintainer – remote login is only allowed for the maintainer.

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

PermitEmptyPasswords no --using of empty passwords is prohibited.

X11 forwarding no – X11 protocol traffic is not allowed because it is not used by the server.

AddressFamily inet – IPv4 is allowed, IPv6 is disabled.

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

```
root@debian:/# systemctl restart sshd
root@debian:/# systemctl status sshd
root@debian:/# ss -Htulp | grep ssh
```

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 did not not append 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:

```
user@server:~$ ssh maintainer@debian.localdomain
maintainer@debian.localdomain's password:
Last login: Sun Aug 11 11:20:37 2019 from 192.168.7.1
```

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.

```
maintainer@debian:~$ logout
Connection to debian.localdomain closed.
user@server:~$ ssh root@debian.localdomain
root@debian.localdomain's password:
Permission denied, please try again.
root@debian.localdomain's password:
Permission denied, please try again.
root@debian.localdomain's password:
root@debian.localdomain: Permission denied (password,keyboard-interactive).
user@server:~$
```

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

If you try to log in from a host other than the allowed one, and the firewall of the server in question is disabled, you would get a message:

```
user@workstation:~$ ssh maintainer@debian.localdomain
ssh_exchange_identification: read: Connection reset by peer
```

as a result of access control lists using. If the firewall is enabled the client receives a different message:

```
user@workstation:~$ ssh maintainer@debian.localdomain
ssh: connect to host debian.localdomain port 22: Connection timed out
```

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

One of the ways to check for automatic restarting the service is shown in the following terminal session:

```
root@debian:/# date
Bc abr 11 12:19:18 MSK 2019
root@debian:/# ps -ef | grep ssh
maintai+  519      1  0 11:20 ?          00:00:01 sshd: maintainer@pts/0
root      852      1  0 12:14 ?          00:00:00 /usr/sbin/sshd -D
root      873    535  0 12:19 pts/0      00:00:00 grep ssh
root@debian:/# kill -9 852
root@debian:/# ps -ef | grep ssh
maintai+  519      1  0 11:20 ?          00:00:01 sshd: maintainer@pts/0
root      875    535  0 12:19 pts/0      00:00:00 grep ssh
root@debian:/# systemctl status sshd
• ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
  Drop-In: /etc/systemd/system/ssh.service.d
           └─local.conf
  Active: activating (auto-restart) (Result: signal) since Sun 2019-08-11 12:19:38
  MSK; 22s ago
    Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 851 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
  Process: 852 ExecStart=/usr/sbin/sshd -D $SSHD_OPTS (code=killed, signal=KILL)
 Main PID: 852 (code=killed, signal=KILL)
    Tasks: 2 (limit: 1149)
   Memory: 7.8M
    CGroup: /system.slice/ssh.service
            └─519 sshd: maintainer@pts/0
              └─520 -bash

abr 11 12:19:38 debian systemd[1]: ssh.service: Main process exited, code=killed,
status=9/KILL
abr 11 12:19:38 debian systemd[1]: ssh.service: Failed with result 'signal'.
root@debian:/# date
Bc abr 11 12:20:18 MSK 2019
root@debian:/# systemctl status sshd
• ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
  Drop-In: /etc/systemd/system/ssh.service.d
           └─local.conf
  Active: active (running) since Sun 2019-08-11 12:20:09 MSK; 12s ago
    Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 878 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 879 (sshd)
    Tasks: 3 (limit: 1149)
```

```
Memory: 8.7M
CGroup: /system.slice/ssh.service
├─519 sshd: maintainer@pts/0
├─520 -bash
└─879 /usr/sbin/sshd -D
```

```
abr 11 12:20:09 debian systemd[1]: This usually indicates unclean termination of a
previous run, or service implementation deficiencies.
abr 11 12:20:09 debian systemd[1]: ssh.service: Found left-over process 520 (bash) in
control group while starting unit. Ignoring.
abr 11 12:20:09 debian systemd[1]: This usually indicates unclean termination of a
previous run, or service implementation deficiencies.
abr 11 12:20:09 debian systemd[1]: Starting OpenBSD Secure Shell server...
abr 11 12:20:09 debian systemd[1]: ssh.service: Found left-over process 519 (sshd) in
control group while starting unit. Ignoring.
abr 11 12:20:09 debian systemd[1]: This usually indicates unclean termination of a
previous run, or service implementation deficiencies.
abr 11 12:20:09 debian systemd[1]: ssh.service: Found left-over process 520 (bash) in
control group while starting unit. Ignoring.
abr 11 12:20:09 debian systemd[1]: This usually indicates unclean termination of a
previous run, or service implementation deficiencies.
abr 11 12:20:09 debian sshd[879]: Server listening on 0.0.0.0 port 22.
abr 11 12:20:09 debian systemd[1]: Started OpenBSD Secure Shell server.
root@debian:/#
```

Here, the system date and time are displayed first, then a list of running processes in the system is obtained and filtered by ssh mention. As you can see from this list, the system is running an sshd process with pid 852. The next command kills this process. It is further demonstrated that the process is indeed killed and that this led to the fail of the relevant service. After 40 seconds (the date and time are displayed again) , the service is successfully restarted.

Installing and configuring lighttpd

You can install the lighttpd web server using the following commands

```
root@debian:/# apt-get update
root@debian:/# apt-get install lighttpd
```

On the system in question the web server requires a minimal set of functions: provide http access to multiple directories. No interpreted programming languages or scripts are used at all, no CMS is used for content management (at least not related to a web server), and no DBMS is used.

The lighttpd web server was chosen because it is stable enough and can implement the required functionality, its resource consumption is lower than that of the more functional apache and nginx web servers.

Configuration

When configuring the web server, you can use Debian-specific presets in the **/etc/lighttpd/conf-available** and **/etc/lighttpd/conf-enabled** directories. However, in this case the web server settings are simple and static, which makes it possible to create a simple configuration file without resorting to mechanisms aimed at more complex and flexible cases of using the web server.

Thus, all the parameters of the web server are specified in a single configuration file **/etc/lighttpd/lighttpd.conf** . Some of the lines there are templates from developers, and some

were created manually to implement specific functionality of the task in question. The final configuration file looks like this (comments are excluded from the file):

```
server.modules = (
    "mod_indexfile",
    "mod_access",
    "mod_alias",
    "mod_redirect",
    "mod_compress",
    "mod_dirlisting",
    "mod_staticfile"
)

server.document-root      = "/var/www"
server.errorlog            = "/var/log/lighttpd/error.log"
server.pid-file           = "/var/run/lighttpd.pid"
server.username           = "www-data"
server.groupname          = "www-data"
server.port               = 80
server.dir-listing        = "enable"
server.kbytes-per-second  = 4096
server.max-connections    = 128
server.max-fds            = 4096

server.http-parseopts = (
    "header-strict"      => "enable",
    "host-strict"        => "enable",
    "host-normalize"     => "enable",
    "url-normalize-unreserved"=> "enable",
    "url-normalize-required" => "enable",
    "url-ctrls-reject"   => "enable",
    "url-path-2f-decode" => "enable",
    "url-path-dotseg-remove" => "enable",
)

index-file.names          = ( "index.php", "index.html" )
url.access-deny           = ( "~", ".inc" )
static-file.exclude-extensions = ( ".php", ".pl", ".fcgi" )
dir-listing.encoding      = "utf-8"
dir-listing.exclude       = ( "lost\+found" )
compress.cache-dir        = "/var/cache/lighttpd/compress/"
compress.filetype         = ( "application/javascript", "text/css",
"text/html", "text/plain" )

include_shell "/usr/share/lighttpd/create-mime.conf.pl"
```

When configuring a web server, you should make the following comments:

The server accepts connections on port 80 (http), no more than 128 connections at a time, the total bandwidth is limited to 4 MB/s (32 Mbit/s), and the web server is allowed to open no more than 4096 file descriptors at a time. The server also supports displaying the contents of directories in UTF 8 encoding, with the exception of the lost+found file system service directory. Web server operation over IPv6 is disabled by deleting the following line from the default configuration file:

```
include_shell "/usr/share/lighttpd/use-ipv6.pl " + server.port
```

The parameters for parsing http requests have retained their default values.

For proper operation of repositories and the web server, you should also adhere to a specific directory structure inside the root folder of the web server **/var/www**:

```
/var/www
├── buster
├── lost+found
├── ppa
└── security
```

However, the html subdirectory was deleted from this folder along with the stub page as unnecessary.

Fail protection (auto-resume)

Similarly ssh, the web server automatically restarting in case of crash is configured by the developers of the operating system distribution.

Verification of performance and stability

To check the status of the web server, use the command

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

or a command

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

which is, in fact, a wrapper of the above.

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

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

To check the correctness of the lighttpd configuration file, use the command

```
root@debian:/etc/lighttpd# /usr/sbin/lighttpd -tt -f /etc/lighttpd/lighttpd.conf
```

The method for checking automatic renewal is completely similar to ssh, so only the corresponding fragment of the terminal session is shown here:

```
root@debian:/# date
Чт авг 15 12:57:07 MSK 2019
root@debian:/# ps -ef | grep lighttpd
www-data 1404      1   0 12:50 ?                00:00:00 /usr/sbin/lighttpd -D -f
/etc/lighttpd/lighttpd.conf
root     1413    505   0 12:57 pts/0        00:00:00 grep lighttpd
root@debian:/# kill -9 1404
root@debian:/# ps -ef | grep lighttpd
root     1416    505   0 12:57 pts/0        00:00:00 grep lighttpd
root@debian:/# systemctl status lighttpd
• lighttpd.service - Lighttpd Daemon
   Loaded: loaded (/lib/systemd/system/lighttpd.service; enabled; vendor preset:
   enabled)
   Drop-In: /etc/systemd/system/lighttpd.service.d
            └─local.conf
   Active: activating (auto-restart) (Result: signal) since Thu 2019-08-15 12:57:35
   MSK; 14s ago
   Process: 1401 ExecStartPre=/usr/sbin/lighttpd -tt -f /etc/lighttpd/lighttpd.conf
   (code=exited, status=0/SUCCESS)
```

```
Process: 1404 ExecStart=/usr/sbin/lighttpd -D -f /etc/lighttpd/lighttpd.conf
(code=killed, signal=KILL)
Main PID: 1404 (code=killed, signal=KILL)
```

```
abr 15 12:57:35 debian systemd[1]: lighttpd.service: Main process exited,
code=killed, status=9/KILL
abr 15 12:57:35 debian systemd[1]: lighttpd.service: Failed with result 'signal'.
```

```
root@debian:/# ps -ef | grep lighttpd
```

```
www-data 1422 1 0 12:58 ? 00:00:00 /usr/sbin/lighttpd -D -f
/etc/lighttpd/lighttpd.conf
```

```
root 1426 505 0 12:58 pts/0 00:00:00 grep lighttpd
```

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

```
• lighttpd.service - Lighttpd Daemon
  Loaded: loaded (/lib/systemd/system/lighttpd.service; enabled; vendor preset:
enabled)
  Drop-In: /etc/systemd/system/lighttpd.service.d
           └─local.conf
  Active: active (running) since Thu 2019-08-15 12:58:05 MSK; 26s ago
  Process: 1419 ExecStartPre=/usr/sbin/lighttpd -tt -f /etc/lighttpd/lighttpd.conf
(code=exited, status=0/SUCCESS)
  Main PID: 1422 (lighttpd)
  Tasks: 1 (limit: 1149)
  Memory: 1.3M
  CGroup: /system.slice/lighttpd.service
          └─1422 /usr/sbin/lighttpd -D -f /etc/lighttpd/lighttpd.conf
```

```
abr 15 12:58:05 debian systemd[1]: Starting Lighttpd Daemon...
```

```
abr 15 12:58:05 debian systemd[1]: Started Lighttpd Daemon.
```

```
root@debian:/#
```

Creating a personal repository

First of all you need to install the reprepro package:

```
root@debian:/# apt-get update
```

```
root@debian:/# apt-get install reprepro
```

Since the repository must be managed on behalf of the maintainer, the root directories of the repositories must belong to this user. To do this, run the following commands:

```
root@debian:/# chown -R maintainer:maintainer /var/www/buster
```

```
root@debian:/# chown -R maintainer:maintainer /var/www/security
```

```
root@debian:/# chown -R maintainer:maintainer /var/www/ppa
```

For more information about access rights, see the next section, but then you should create files and subdirectories as maintainer.

Then go to the root directory of the repository being created (**/var/www/ppa**). Inside it, without paying attention to the web server files, you need to create a conf directory, go to it, and create a **distributions** file with the following contents:

```
Description: repository with custom packages
Origin: debian
Suite: stable
AlsoAcceptFor: testing unstable experimental
Codename: buster
Version: 10.0
Architectures: amd64
Components: main contrib
```


Then, go back to the root directory of the repository and run the following commands:

```
maintainer@debian:/var/www/ppa$ reprepro export
maintainer@debian:/var/www/ppa$ reprepro createsymlinks
```

After that, the **/db**, **/dists**, and **/pool** subdirectories will appear in the root directory of the repository. The repository is created, now you need to create a digital signature of the repository.

First of all, a gpg key pair is created in order to sign the repository (these actions should be performed on behalf of the maintainer user):

```
maintainer@debian:/var/www/ppa$ gpg --gen-key
```

During the key creation process, you will need to answer several questions. most of them (about the key type and its length) can be left with the default answers (RSA 3072). The key expiration date can be omitted (as part of this task).

Then export the public part of the key to a file:

```
maintainer@debian:/var/www/ppa$ gpg --armor --export > repository_key.asc
```

You must then sign the repository. To do this, go to the release file in the repository (in this case, **/var/www/ppa/dists/buster/Release**) and sign it with the received key:

```
maintainer@debian:/var/www/ppa$ cd /var/www/ppa/dists/buster
maintainer@debian:/var/www/ppa/dists/buster$ gpg --armor -o Release.gpg -sb Release
```

As a result, the **Release.gpg** file will appear in the same folder. This is the digital signature of the repository. Each time the repository content changes, the administrator must re-sign the Release file (s) with the same key (in this case, on behalf of the maintainer, the private part of the key is stored in the home directory **/home/maintainer/**). The information needed to create and use a key should be stored in a secure place, preferably on a removable media separate from the system. In this case, even after gaining full control of the system, the attacker will not be able to place their packages in the personal repository and compromise the repository's clients.

Note that you only need to sign the main Release file of each repository. Content of the personal repository at the time of writing this document is:

```
/var/www/ppa
├── [drwxr-xr-x] conf
│   └── [-rw-r--r--] distributions
├── [drwxr-xr-x] db
│   ├── [-rw-r--r--] checksums.db
│   ├── [-rw-r--r--] contents.cache.db
│   ├── [-rw-r--r--] packages.db
│   ├── [-rw-r--r--] references.db
│   ├── [-rw-r--r--] release.caches.db
│   └── [-rw-r--r--] version
├── [drwxr-xr-x] dists
│   ├── [drwxr-xr-x] buster
│   │   ├── [drwxr-xr-x] contrib
│   │   │   ├── [drwxr-xr-x] binary-amd64
│   │   │   │   ├── [-rw-r--r--] Packages
│   │   │   │   ├── [-rw-r--r--] Packages.gz
│   │   │   └── [-rw-r--r--] Release
│   │   └── [drwxr-xr-x] main
│   │       ├── [drwxr-xr-x] binary-amd64
│   │       │   ├── [-rw-r--r--] Packages
│   │       │   └── [-rw-r--r--] Packages.gz
```

```

├── ┌── ┌── [-rw-r--r--] Release
│   │   └── [-rw-r--r--] Release
│   │       └── [-rw-r--r--] Release.gpg
│   └── [lrwxrwxrwx] stable -> buster
└── [drwxr-xr-x] pool
    └── [drwxr-xr-x] main
        ├── [drwxr-xr-x] d
        │   └── [drwxr-xr-x] dirclean
        │       └── [-rw-r--r--] dirclean_1.0-1_all.deb
        ├── [drwxr-xr-x] libv
        │   ├── [drwxr-xr-x] libvcr
        │   │   └── [-rw-r--r--] libvcr_1.2-1_amd64.deb
        │   ├── [drwxr-xr-x] libvcr-dev
        │   │   └── [-rw-r--r--] libvcr-dev_1.2-1_amd64.deb
        ├── [drwxr-xr-x] q
        │   └── [drwxr-xr-x] qsquidclassroom
        │       └── [-rw-r--r--] qsquidclassroom_1.1-1_amd64.deb
        ├── [drwxr-xr-x] r
        │   ├── [drwxr-xr-x] restore-settings
        │   │   └── [-rw-r--r--] restore-settings_1.0-3_all.deb
        └── [drwxr-xr-x] t
            ├── [drwxr-xr-x] tux-user-avatars
            │   └── [-rw-r--r--] tux-user-avatars_1.0-1_all.deb

```

Here, the repository contains three files named Release, and the main file is the one that is closest to the repository root. It is signed, i.e. the Release.gpg file is located next to it.

Repository content protection

When mirroring a repository, i.e. moving a large volume of files between different file systems that do not necessarily support a common set of attributes, or when managing a personal repository, the files and directories access attributes within repositories may differ from the desired ones. Checking and installing the necessary permissions manually, even for a small repository, is extremely time-consuming. It is better to develop and place a small script in the system:

```

root@debian:~# cat > /usr/local/bin/repattr
#!/bin/bash
#Setting comandline arguments smart names
REPROOT=$1
REPUSER=$2
REPGROUP=$3
#If no repository root given or it is not a directory then exit with errorcode 1
if [ ! -d "$REPROOT" ]
then
    exit 1
fi
#If no user given then assume root
if [ -z "$REPUSER" ]
then
    REPUSER=root
fi
#If no group given then assume root
if [ -z "$REPGROUP" ]
then
    REPGROUP=root
fi
#Changing owner for all files and directories
chown -R $REPUSER:$REPGROUP $REPROOT

```

```
#Setting minimal possible permissions to both files and directories
chmod -R 0644 $REPR00T
#Allowing to enter into directories
tree -dfRi $REPR00T | grep "/" | grep -v "\->" | xargs -I dir chmod 0755 dir
#Exiting
exit 0
^D
root@debian:~# chmod a+x /usr/local/bin/repattr
```

The repository is functional with the following access attributes:

- the owner of all files and directories in the repository is an unprivileged user maintainer;
- all directories have full permissions for the owner and read and execute (open) permissions for members of his/her group and all others (0755);
- all files in directories have read and write accessible for the owner and read-only for members of the owner's group and everyone else (0644).

Of course, you should run this script with sufficient permissions for it to work, i.e. in the vast majority of cases on behalf of the superuser.

Running the script to correctly configure access rights and the maintainer owning for a personal repository should look like this:

```
root@debian:~# repattr /var/www/ppa/ maintainer maintainer
```

For repositories with a large number of packages, the script may take a considerable time to work. You can check the results of the script using the following command:

```
root@debian:~# tree -Rugp /var/www/ppa/
```

However, you should use it wisely, because for large repositories its output will also be large and will take some time.

About additional services

Usually, when configuring a web server, which is the repository server in question, we consider installing a number of additional services to ensure its performance and stability. These services include a database server, a network time service, anti-attack tools, log file management, backup and synchronization. However, each such service consumes system resources to some extent. Therefore, based on the principle of reasonable sufficiency, the role of each such service should be evaluated and a decision should be made on whether to use it.

Database server

Based on the device and mechanisms of the repositories, as well as taking into account the lack of additional web resources on the server in question, it is clear that the DBMS is not used on this server, i.e. it does not need to be installed.

Password protection (fail2ban)

Since ssh remains the only login mechanism due to the nature of the server and the lack of authorization in the system via a web 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 entries), remote login is allowed only for an unprivileged user whose account is protected by a strong password, and increasing privileges requires knowledge of also strong superuser password. In this case, the only host that allows remote login is also the virtualization server that hosts the virtual server in question. In these circumstances, an attack on the server by trying to match passwords and accounts is only possible if the attacker has control of the virtualization server itself. But in this case, intruder can get full control of the virtual server without logging into it. Because of the above, using fail2ban is excessive and leads only to a waste of system resources.

Log rotation (logrotate)

System logs can serve as means of attacking the server by filling its disk space and thus paralyzing its operation. This is opposed by the log file rotation system.

When the settings are set, lighttpd only keeps an error log `/var/log/lighttpd/error.log`. The error log also contains information about starting and stopping the web server. It is acceptable not to consider this log as the target of an attack, because if the attacker was able to ensure the avalanche growth of this log, then he/she already has enough authority in the system and no longer needs to attack the logs. The access log, on the other hand, is kept in detail and may be the target of such attack. At the same time, in normal operation, records of this log, given the role and location of the server in the network, are not particularly valuable. Therefore, the access log is not kept in this case.

The same reasoning applies to ssh. When logging in using this protocol, 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 the virtualization server where the virtual repository server itself is located, but this is pointless, since controlling the host system, the attacker has all the possibilities to get the guest's data without hacking the guest (of course, if the virtual disk encryption is not used, in this case, encryption does not make sense). Stopping the guest's work will not cause any problems for the attacker.

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

Network time service (ntp)

The network time service is also redundant for this server: since the server is virtual and no physical time source is connected to it, it uses the system clock of the virtualization server on which it is located. This solution can be considered reliable, because a compromised virtualization server leaves no chance for the virtual machines located on it.

Mail service

A mail service, even a local one, can become a target for attack: for example, by exploiting web server vulnerabilities, 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 the system in question, the mail components were not selected during the installation stage. Also no installed services are dependent on them.

Regarding auxiliary email tasks such as informing the system administrator about service or hardware failures, it can be argued that this functionality is redundant on the server in question. Since the server is virtual, hardware failures on it are either the result of unstable hypervisor operation, or the result of failures of the physical hardware of the virtualization server. In any case, the virtualization server itself will inform the system administrator, not the virtual server. At the same time, the virtual server services are configured to automatically resume operation in case of failures and do not require manual intervention.

Installing and configuring the network filter (nftables)

By default, the iptables interface of the netfilter core network filter was installed in the system. To delete it and install the nftables interface that replaced it run the following commands:

```
root@debian:/# apt-get purge iptables
root@debian:/# apt-get install nftables
```

The next step is to determine the method for creating a set of network filter rules. There are two possible approaches: configure all the rules at the same time at the system boot stage before starting the network interfaces, and run a minimal, wireframe set of rules at the same stage, then add rules before starting each interface and delete them when it stops. The second approach is more flexible and convenient in systems with complex dynamically changing routing schemes, channel redundancy, and so on. There are no such features on the system under consideration, so atomically loading all the rules at once is the most balanced decision.

The systemd Manager's nftables.service protection against unexpected service crashes is redundant: this service performs a firewall rule loading operation and terminates immediately after it finishes, as you can see from the output of the following command:

```
root@debian:/# systemctl status nftables
• nftables.service - nftables
   Loaded: loaded (/lib/systemd/system/nftables.service; enabled; vendor preset:
   enabled)
   Active: active (exited) since Thu 2019-08-15 10:18:21 MSK; 10h ago
     Docs: man:nft(8)
           http://wiki.nftables.org
  Main PID: 185 (code=exited, status=0/SUCCESS)
    Tasks: 0 (limit: 1149)
   Memory: 0B
    CGroup: /system.slice/nftables.service
```

```
abr 15 10:18:21 localhost systemd[1]: Started nftables.
Warning: Journal has been rotated since unit was started. Log output is incomplete or
unavailable.
```

```
root@debian:/#
```

Open port

The network filter should not block ports used by server components. To get a list of these ports, use the following commands

```
root@debian:/# ss -ltn inet
```

Netid	State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port
udp	UNCONN	0	0	0.0.0.0:bootpc	0.0.0.0:*
tcp	LISTEN	0	128	0.0.0.0:ssh	0.0.0.0:*
tcp	LISTEN	0	128	0.0.0.0:http	0.0.0.0:*

```

root@debian:/# ss -ltn inet6
Netid      State      Recv-Q     Send-Q     Local Address:Port      Peer Address:Port
root@debian:/#

```

From the output of these commands, it follows that the services provided by the system listen only to IPv4 port ssh (22) and http (80). There are no open IPv6 ports.

Network environment

The schematic diagram of the virtual server location in the network is shown in the figure:

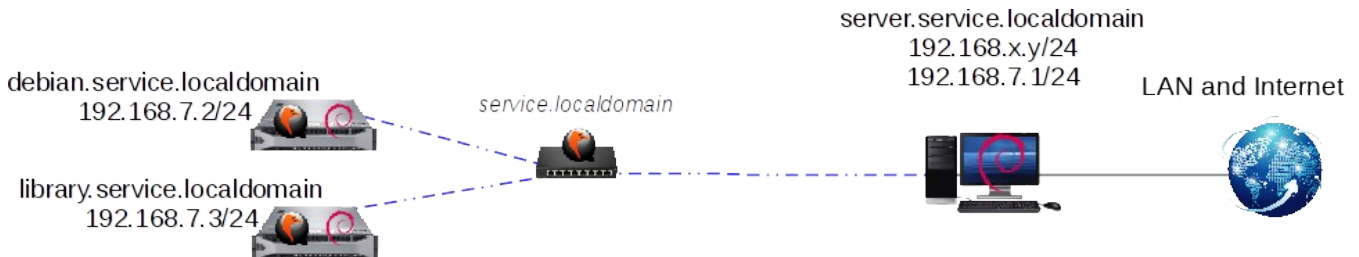


Figure 1: network diagram

This diagram shows that the repository server is connected to the service virtual network of the local domain, to which other virtual servers (such as library) are connected too. Communication with external networks (LAN and Internet) is only possible via the virtualization server. However, the repository can be used by any node represented in the diagram: any computer from the LAN (or even the Internet, if the LAN allows it), the virtualization server itself, or any other virtual server. At the same time, ssh connections are only allowed from the virtualization server. Among other things, you should consider the ability of the server to respond to certain ICMP requests, which is very convenient for quickly checking the availability of the server. However, the number of such requests per unit of time should be limited so that they cannot become a tool in the hands of an attacker to generate useless traffic. You should also limit the number of connection requests for other protocols. Optimal restrictions can only be identified during the server's operational phase, but the following values are reasonable at the initial stage:

- maximum of one request per second for ssh connection
- maximum of ten icmp requests per second
- maximum of one hundred requests per second for http connection

In addition, all IPv6 packets received or generated by the server are completely cut off by the firewall.

The structure of the network filter

Summing up all the above, the following script **/etc/nftables.conf** is developed for atomic loading of firewall rules:

```

#!/usr/sbin/nft -f

flush ruleset

table ip filter {
    chain input {
        type filter hook input priority 0;
        policy drop;
    }
}

```

```

        # accept any localhost traffic
        iif lo accept
        # accept traffic originating from us
        ct state established,related accept
        # accept traffic from subnet server to local ssh port
        ip saddr 192.168.7.1 ether saddr 52:54:00:2b:1e:a0 tcp dport ssh ct
state new limit rate 1/second accept
        # accept icmp traffic from everywhere
        icmp type echo-request limit rate 10/second accept
        # accept traffic to http port from everywhere
        tcp dport http ct state new limit rate 100/second accept
    }
    chain output {
        type filter hook output priority 0;
        policy accept;
    }
}

table ip6 filter {
    chain input {
        type filter hook input priority 0;
        policy drop;
    }
    chain output {
        type filter hook output 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 which must be accurate to the comments and the pre-cleaning command to 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 content of the **/var/log/daemon.log** file. As an example, the following excerpt from this file (with abbreviations):

```

Aug 16 13:00:46 localhost systemd[1]: Started Apply Kernel Variables.
...
Aug 16 13:00:46 localhost systemd[1]: Starting udev Kernel Device Manager...
Aug 16 13:00:46 localhost systemd[1]: Started udev Kernel Device Manager.
Aug 16 13:00:46 localhost systemd[1]: Started nftables.
Aug 16 13:00:46 localhost systemd[1]: Reached target Network (Pre).
Aug 16 13:00:46 localhost systemd-udevd[221]: Using default interface naming scheme 'v240'.
Aug 16 13:00:46 localhost systemd-udevd[221]: link_config: autonegotiation is unset or enabled, the speed and duplex are not writable.
...
Aug 16 13:00:46 localhost systemd-udevd[220]: link_config: autonegotiation is unset or enabled, the speed and duplex are not writable.
...
Aug 16 13:00:46 localhost systemd[1]: Reached target Basic System.
...
Aug 16 13:00:46 localhost systemd[1]: Starting Raise network interfaces...

```

```
...
Aug 16 13:00:51 localhost ifup[323]: bound to 192.168.7.2 -- renewal in 2147483648
seconds.
Aug 16 13:00:51 localhost dhclient[344]: bound to 192.168.7.2 -- renewal in
2147483648 seconds.
Aug 16 13:00:51 localhost systemd[1]: Started Raise network interfaces.
Aug 16 13:00:51 localhost systemd[1]: Reached target Network.
```

You can check the network filter using the following test program:

№	Action	Expected results
1	Ssh login from any workstation in the LAN	Login impossible
2	Ssh login from other virtual machine in the same virtual network	Login impossible
3	Obtaining file from LAN workstation via http	File obtained
4	Obtaining file from other virtual machine via http	File obtained
5	Network scanning from LAN workstation	Port 80 opened
6	Network scanning from other virtual machine	Port 80 opened

The following fragment of the terminal session shows the server being checked by the workstation:

```
root@workstation:/# ssh maintainer@debian.service.localdomain
ssh: connect to host debian.service.localdomain port 22: Connection timed out
root@workstation:/# wget http://debian.service.localdomain/ppa/dists/buster/Release
--2019-08-19 20:29:30-- http://debian.service.localdomain/ppa/dists/buster/Release
Распознаётся debian.service.localdomain (debian.service.localdomain)... 192.168.7.2
Подключение к debian.service.localdomain (debian.service.localdomain)|
192.168.7.2|:80... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: 1655 (1,6K) [application/octet-stream]
Сохранение в: «Release»

Release                               100%[=====>]  1,62K  --.-KB/s  за 0s

2019-08-19 20:29:30 (227 MB/s) - «Release» сохранён [1655/1655]

root@workstation:/# rm Release
root@workstation:/# nmap -sS 192.168.7.2
Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:30 GMT-3

Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00052s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 6.14 seconds
root@workstation:/# nmap -sT 192.168.7.2

Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:31 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00072s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 5.66 seconds
```



```
root@workstation:/# nmap -sU 192.168.7.2
```

```
Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:31 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00053s latency).
All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered
```

```
Nmap done: 1 IP address (1 host up) scanned in 22.35 seconds
```

```
root@workstation:/# nmap -sN 192.168.7.2
```

```
Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:32 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00053s latency).
All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered
```

```
Nmap done: 1 IP address (1 host up) scanned in 22.34 seconds
```

```
root@workstation:/# nmap -sF 192.168.7.2
```

```
Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:32 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00051s latency).
All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered
```

```
Nmap done: 1 IP address (1 host up) scanned in 22.33 seconds
```

```
root@workstation:/# nmap -sX 192.168.7.2
```

```
Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:33 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00059s latency).
All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered
```

```
Nmap done: 1 IP address (1 host up) scanned in 22.31 seconds
```

```
root@workstation:/# nmap -sA 192.168.7.2
```

```
Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:33 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00051s latency).
Not shown: 999 filtered ports
PORT      STATE      SERVICE
80/tcp    unfiltered http
```

```
Nmap done: 1 IP address (1 host up) scanned in 6.08 seconds
```

```
root@workstation:/# nmap -sW 192.168.7.2
```

```
Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:34 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00053s latency).
Not shown: 999 filtered ports
PORT      STATE      SERVICE
80/tcp    closed    http
```

```
Nmap done: 1 IP address (1 host up) scanned in 19.79 seconds
```

```
root@workstation:/# nmap -sM 192.168.7.2
```

```
Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:34 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00058s latency).
All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered
```

```
Nmap done: 1 IP address (1 host up) scanned in 22.30 seconds
```

```
root@workstation:/# nmap -sO 192.168.7.2
```

```
Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:35 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.022s latency).
Not shown: 254 open|filtered protocols
PROTOCOL STATE SERVICE
1         open  icmp
6         open  tcp
```

```
Nmap done: 1 IP address (1 host up) scanned in 9.71 seconds
root@workstation:/# nmap -sV -sS 192.168.7.2
```

```
Starting Nmap 6.47 ( http://nmap.org ) at 2019-08-19 20:35 GMT-3
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (0.00053s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE VERSION
80/tcp    open  http    lighttpd 1.4.53
```

```
Service detection performed. Please report any incorrect results at
http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 24.21 seconds
root@workstation:/#
```

Summing up the check, it should be noted that it was possible to find out the presence of an open http port on which the lighttpd web server version 1.4.53 runs, which successfully provided the file on request, and the server's responsiveness to icmp requests was also confirmed. Thus, the results obtained do not contradict the expected results and the verification can be considered successful.

The following fragment of the terminal session demonstrates checking the server from another virtual machine:

```
root@library:/# ssh maintainer@debian.service.localdomain
ssh: connect to host debian.service.localdomain port 22: Connection timed out
root@library:/# wget http://debian.service.localdomain/ppa/dists/buster/Release
--2019-08-19 21:03:27-- http://debian.service.localdomain/ppa/dists/buster/Release
Распознаётся debian.service.localdomain (debian.service.localdomain)... 192.168.7.2
Подключение к debian.service.localdomain (debian.service.localdomain)|
192.168.7.2|:80... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: 1655 (1,6K) [application/octet-stream]
Сохранение в: «Release»
```

```
Release          100%[=====>] 1,62K  --.-KB/s   за 0s
```

```
2019-08-19 21:03:27 (271 MB/s) - «Release» сохранён [1655/1655]
```

```
root@library:/# rm Release
root@library:/# nmap -sS 192.168.7.2
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-19 21:04 MSK
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (-0.078s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)
```

```
Nmap done: 1 IP address (1 host up) scanned in 9.35 seconds
root@library:/# nmap -sT 192.168.7.2
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-19 21:04 MSK
```

Nmap scan report for debian.localdomain (192.168.7.2)

Host is up (-0.13s latency).

Not shown: 999 filtered ports

PORT	STATE	SERVICE
80/tcp	open	http

MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 4.99 seconds

root@library:/# nmap -sU 192.168.7.2

Starting Nmap 7.70 (<https://nmap.org>) at 2019-08-19 21:04 MSK

Nmap scan report for debian.localdomain (192.168.7.2)

Host is up (-0.20s latency).

All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered

MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 21.35 seconds

root@library:/# nmap -sN 192.168.7.2

Starting Nmap 7.70 (<https://nmap.org>) at 2019-08-19 21:05 MSK

Nmap scan report for debian.localdomain (192.168.7.2)

Host is up (-0.20s latency).

All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered

MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 21.35 seconds

root@library:/# nmap -sF 192.168.7.2

Starting Nmap 7.70 (<https://nmap.org>) at 2019-08-19 21:06 MSK

Nmap scan report for debian.localdomain (192.168.7.2)

Host is up (-0.20s latency).

All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered

MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 21.35 seconds

root@library:/# nmap -sX 192.168.7.2

Starting Nmap 7.70 (<https://nmap.org>) at 2019-08-19 21:06 MSK

Nmap scan report for debian.localdomain (192.168.7.2)

Host is up (-0.19s latency).

All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered

MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 21.35 seconds

root@library:/# nmap -sA 192.168.7.2

Starting Nmap 7.70 (<https://nmap.org>) at 2019-08-19 21:07 MSK

Nmap scan report for debian.localdomain (192.168.7.2)

Host is up (-0.060s latency).

Not shown: 999 filtered ports

PORT	STATE	SERVICE
80/tcp	unfiltered	http

MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 10.68 seconds

root@library:/# nmap -sW 192.168.7.2

Starting Nmap 7.70 (<https://nmap.org>) at 2019-08-19 21:07 MSK

Nmap scan report for debian.localdomain (192.168.7.2)

Host is up (-0.078s latency).

Not shown: 999 filtered ports

PORT	STATE	SERVICE
80/tcp	closed	http

MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 9.35 seconds

root@library:/# nmap -sM 192.168.7.2

Starting Nmap 7.70 (<https://nmap.org>) at 2019-08-19 21:07 MSK

```
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (-0.20s latency).
All 1000 scanned ports on debian.localdomain (192.168.7.2) are open|filtered
MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)
```

```
Nmap done: 1 IP address (1 host up) scanned in 21.34 seconds
root@library:/# nmap -sO 192.168.7.2
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-19 21:08 MSK
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (-0.12s latency).
Not shown: 254 open|filtered protocols
PROTOCOL STATE SERVICE
1          open  icmp
6          open  tcp
MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)
```

```
Nmap done: 1 IP address (1 host up) scanned in 6.29 seconds
root@library:/# nmap -sV -sS 192.168.7.2
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-19 21:08 MSK
Nmap scan report for debian.localdomain (192.168.7.2)
Host is up (-0.060s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE VERSION
80/tcp    open  http    lighttpd 1.4.53
MAC Address: 52:54:00:79:69:B1 (QEMU virtual NIC)
```

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

```
Nmap done: 1 IP address (1 host up) scanned in 19.14 seconds
root@library:/#
```

Summing up the results of this check, it should be noted that it was also possible to find out the presence of an open http port on which the lighttpd web server version 1.4.53 is running, which successfully provided the file on request, and confirm the server's responsiveness to icmp requests. In addition, we were able to find out the MAC address of the server, which is not surprising, since they are located on the same local network. Thus, the results obtained do not contradict the expected results and the verification can be considered successful.

Completing the installation

At the end of the installation, it makes sense to clear the local package cache and delete the packages that the system no longer needs. You can do this with commands:

```
root@debian:/# apt-get autoremove
root@debian:/# apt-get clean
```

Using a virtual server

Below are brief instructions on how to use the virtual server and the associated tasks.

How to change the virtualization server

We are not considering the mechanism of migration of virtual machines, here discusses only the setup of the virtual server after migration.

If the new host server has a different address of the virtual interface, a different set of allowed hosts is provided for ssh connection, the network settings are generally different (for example, DHCP is not used), then you should change the following configuration files and parameters in them:

- **/etc/hosts.allow** – service access lists;
- **/etc/network/interfaces** – network interfaces, addresses, and static arp entries;
- **/etc/nftables.conf** – network filter settings, network interface names, ip and lladdr addresses.

You should also edit or erase configuration files with ssh service information in the users' home directories:

- **/root/.ssh/known_hosts**
- **/home/maintainer/.ssh/known_hosts**

These files can be sources of valuable information about the network where the virtual server was previously located, and it is highly undesirable for this information to fall into the wrong hands.

Login procedure

To log in remotely, use the ssh protocol and connect only from the virtualization server that hosts the server in question. If you need to connect from a different machine, connect to the virtualization server from that machine, and then connect to the target server from that machine.

Only the maintainer user is allowed to connect. If you need to gain your rights, you should first log in under this user, and then get superuser rights using the su utility.

The command to connect to the server can look like this (up to the current user on the virtualization server and/or how to specify the target server):

```
user@server:/# ssh maintainer@debian.service.localdomain
user@server:/# ssh maintainer@debian.localdomain
user@server:/# ssh maintainer@192.168.7.2
```

How to synchronize with official repositories

To synchronize with official repositories, use the methods used in the Debian community. So instead of using rsync directly, you should use specialized software that minimizes the load on both hosts (local and official repositories) and traffic on the network. For local mirrors at the organization/enterprise level, the debmirror package is one of the recommended ones. In its work it also uses rsync. It should be noted that when working as an unprivileged user, debmirror issues a number of warnings, so in the following instructions this process is started by a superuser.

To synchronize the main repository and the security updates repository just run the following commands:

```
root@debian:/# cd /var/www/buster
root@debian:/var/www/buster# debmirror --nosource --i18n \
--host=mirror.mephi.ru --root=debian --method=http --progress --ignore-release-gpg \
```

```
--dist=buster --arch=amd64 --section=main,contrib,non-free ./
root@debian:/var/www/buster# cd /var/www/security
root@debian:/var/www/security# debmirror --nosource --i18n \
--host=security.debian.org --root=debian-security --method=http --progress \
--ignore-release-gpg --dist=buster/updates --arch=amd64 --section=main,contrib ./
root@debian:/var/www/security# cd /home/maintainer
root@debian:/home/maintainer# repattr /var/www/buster/ maintainer maintainer
root@debian:/home/maintainer# repattr /var/www/security/ maintainer maintainer
```

Since mirroring was performed on behalf of the superuser, maintainer or www-data users may have difficulties accessing the repository content. This is why the repattr script described above should be ran.

Of course, all these commands can be written in the form of a command-line script, for example, like this:

```
maintainer@debian:~$ cat /home/maintainer/updaterepo
#!/bin/bash

DIR=$(pwd)
cd /var/www/buster
debmirror --nosource --i18n --host=mirror.mephi.ru --root=debian --method=http --
progress --ignore-release-gpg --dist=buster --arch=amd64 --section=main,contrib,
non-free ./
cd /var/www/security
debmirror --nosource --i18n --host=security.debian.org --root=debian-security --
method=http --progress --ignore-release-gpg --dist=buster/updates --arch=amd64 --
section=main,contrib ./
cd $DIR
repattr /var/www/buster/ maintainer maintainer
repattr /var/www/security/ maintainer maintainer
```

The above script should, of course, be ran with superuser privileges.

How to manage a personal repository

To add a package to the repository, run the following command:

```
reprepro [-C <category>] includedeb <codename> <package>
```

where <category> is one of the components values,

<codename> – the distribution's code name,

<package> – name of the package file (including the path).

The command should be ran from the root directory of the repository.

Example of commands:

```
maintainer@debian:~$ cd /var/www/ppa && reprepro -C main includedeb buster \
dirclean_1.0-1_all.deb
```

If you need to add several packages at once, the appropriate command is:

```
maintainer@debian:~$ cd /var/www/ppa && reprepro -C main includedeb buster *.deb
```

In other words, the use of file name templates is supported. For example, to add all apache2 packages located in the /tmp/raw_deb/ directory, you can use the command:

```
maintainer@debian:~$ cd /var/www/ppa && reprepro includedeb buster \  
/tmp/raw_deb/apache2*.deb
```

However, if you do not specify a category, the system will automatically determine it by the package content (control file) for each package.

To delete a package from the repository, use the following command:

```
reprepro -C <category> remove <codename> <package>
```

After manipulating packages, you should re-sign the main release files of the repository (not to be confused with the subordinate Release files in the repository sections). In the system under consideration, this file is the only one (**/var/www/ppa/dists/buster/Release**), respectively, the commands take the form:

```
maintainer@debian:/$ cd /var/www/ppa/dists/buster  
maintainer@debian:/var/www/ppa/dists/buster$ gpg --armor -o Release.gpg -sb Release
```

Connecting client to the repositories

To connect a computer to repositories, first enter some or all of the following lines in its **/etc/apt/sources.list** file (up to the domain name, here localdomain):

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

Then you need to register the public part of the key on the client computer (the file **repository_key.asc** obtained when creating a personal repository should be delivered to the client computer in any secure way):

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

To check whether the key was added successfully (to get a list of all registered keys), use the command:

```
root@client:/tmp# apt-key list
```

Finally, you should update the package information on the client and make sure that the repositories are detected, accessible, and their digital signatures are accepted:

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