Lab5 – Hardening

Document your commands or take screenshots. Answer questions in english or finnish. Replace student-id with your own student-id in the labs.

This lab will only use one VM, directly bridged to outside network. Ask your teacher for the VM template path. Get the **Debian** VM template and import it with the name Debian_Hardened. Make sure the VM network is set to bridged and boot up the VM. Default credentials are **root / root66**. Make sure the VM gets an IP address.

```
root@debian:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:09:21:00
          inet addr:192.168.39.154  Bcast:192.168.39.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe09:2100/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:274 errors:0 dropped:0 overruns:0 frame:0
          TX packets:129 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:46671 (45.5 KiB)  TX bytes:23393 (22.8 KiB)
```

- ### Auditing changes (1p)

First install the auditd package:

```
apt update
```

```
root@debian:~# apt update_
```

```
apt install auditd
```

```
root@debian:~# apt install auditd_
```

Also, this Debian VM does not come with sudo installed, so install it:

```
apt install sudo
```

```
root@debian:~# apt install sudo_
```

Then create a rule to watch changes in /etc/passwd (this is where accounts are stored):

```
auditctl -a exit,always -F path=/etc/passwd -F perm=wa
```

```
root@debian:~# auditctl -a exit,always -F path=/etc/passwd -F perm=wa
```

Create also a rule so anytime sudo is run, this gets audited:

```
auditctl -a exit,always -F path=/usr/bin/sudo -F perm=x
```

```
root@debian:~# auditctl -a exit,always -F path=/usr/bin/sudo -F perm=x
```

Now you can try to find events related to the file:

```
ausearch -f /etc/passwd
```

```
root@debian:~# ausearch -f /etc/passwd
<no matches>
```

This should give you no matches as the file has not yet been changed. You can also try to search /usr/bin/sudo. Remember these commands, they will be used later on.

```
root@debian:~# ausearch -f /usr/bin/sudo
<no matches>
```

We don't want to use root for administration so create a separate user for daily administration:

```
useradd -m -s /bin/bash sysadmin
passwd sysadmin
```

```
root@debian:~# useradd -m -s /bin/bash sysadmin
root@debian:~# passwd sysadmin
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Set the password as Qwer-123. Then add the user to sudo group:

```
usermod -G sudo sysadmin
```

```
root@debian:~# usermod -G sudo sysadmin
```

Then login as the user sysadmin and test that he can run some sudo command (for example sudo ifconfig)

Now check the audit events with ausearch and you should see both the user creation and sudo access. The output can be pretty hard to read, but try to find at least the program used to modify passwd-file and the user ID (uid=) that used the sudo. You can check the sysadmins id with either id-command or by looking through the passwd-file. You can also use this uid in ausearch:

```
root@debian:~# su sysadmin
sysadmin@debian:/root$ _
```

```
time->Thu Feb  9 11:54:02 2017
type=PROCTITLE msg=audit(1486634042.677:33): proctitle=7375646F00617573656172636
8002D66002F7573722F62696E2F7375646F
type=PATH msg=audit(1486634042.677:33): item=1 name=(null) inode=393223 dev=fe:0
0 mode=0100755 ouid=0 ogid=0 rdev=00:00 nametype=NORMAL
type=PATH msg=audit(1486634042.677:33): item=0 name="/usr/bin/sudo" inode=150365
 dev=fe:00 mode=0104755 ouid=0 ogid=0 rdev=00:00 nametype=NORMAL
type=CWD msg=audit(1486634042.677:33):  cwd="/root"
type=EXECVE msg=audit(1486634042.677:33): argc=4 a0="sudo" a1="ausearch" a2="-f"
 a3="/usr/bin/sudo"
type=BPRM_FCAPS msg=audit(1486634042.677:33): fver=0 fp=0000000000000000 fi=0000
000000000000 fe=0 old_pp=0000000000000000 old_pi=0000000000000000 old_pe=0000000
000000000 new_pp=0000003ffffffffff new_pi=0000000000000000 new_pe=0000003ffffffff
f
type=SYSCALL msg=audit(1486634042.677:33): arch=c000003e syscall=59 success=yes
exit=0 a0=d2f3a8 a1=d39808 a2=e16008 a3=7ffd8f821ea0 items=2 ppid=2153 pid=2172
auid=0 uid=1000 gid=1000 euid=0 suid=0 fsuid=0 egid=1000 sgid=1000 fsgid=1000 tt
y=tty1 ses=1 comm="sudo" exe="/usr/bin/sudo" key=(null)
```

ausearch -ui <uid of user>

```
sysadmin@debian:/root$ sudo ausearch -ui 1000_
time->Thu Feb  9 11:53:27 2017
type=USER_AUTH msg=audit(1486634007.645:22): pid=2161 uid=1000 auid=0 ses=1 msg=
'op=PAM:authentication acct="sysadmin" exe="/usr/bin/sudo" hostname=? addr=? ter
minal=/dev/tty1 res=success'
----
time->Thu Feb  9 11:53:27 2017
type=USER_ACCT msg=audit(1486634007.645:23): pid=2161 uid=1000 auid=0 ses=1 msg=
'op=PAM:accounting acct="sysadmin" exe="/usr/bin/sudo" hostname=? addr=? termina
l=/dev/tty1 res=success'
----
time->Thu Feb  9 11:53:27 2017
type=USER_CMD msg=audit(1486634007.649:24): pid=2161 uid=1000 auid=0 ses=1 msg='
cwd="/root" cmd=617573656172636820 2D66202F6574632F706173737764 terminal=tty1 res
=success'
----
time->Thu Feb  9 11:54:02 2017
type=USER_CMD msg=audit(1486634042.681:34): pid=2172 uid=1000 auid=0 ses=1 msg='
cwd="/root" cmd=617573656172636820 2D66202F7573722F62696E2F7375646F terminal=tty1
 res=success'
----
time->Thu Feb  9 11:55:17 2017
type=USER_CMD msg=audit(1486634117.777:40): pid=2174 uid=1000 auid=0 ses=1 msg='
cwd="/root" cmd=617573656172636820 2D75692031303030 terminal=tty1 res=success'
```

- **SSH logins (1p)**

Since this machine had no other account than root, it is safe to assume it has root account allowed in the SSH config. Modify /etc/ssh/sshd_config and change **PermitRootLogin** to **no**

```
  GNU nano 2.2.6          File: /etc/ssh/sshd_config

SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin no
StrictModes yes
```

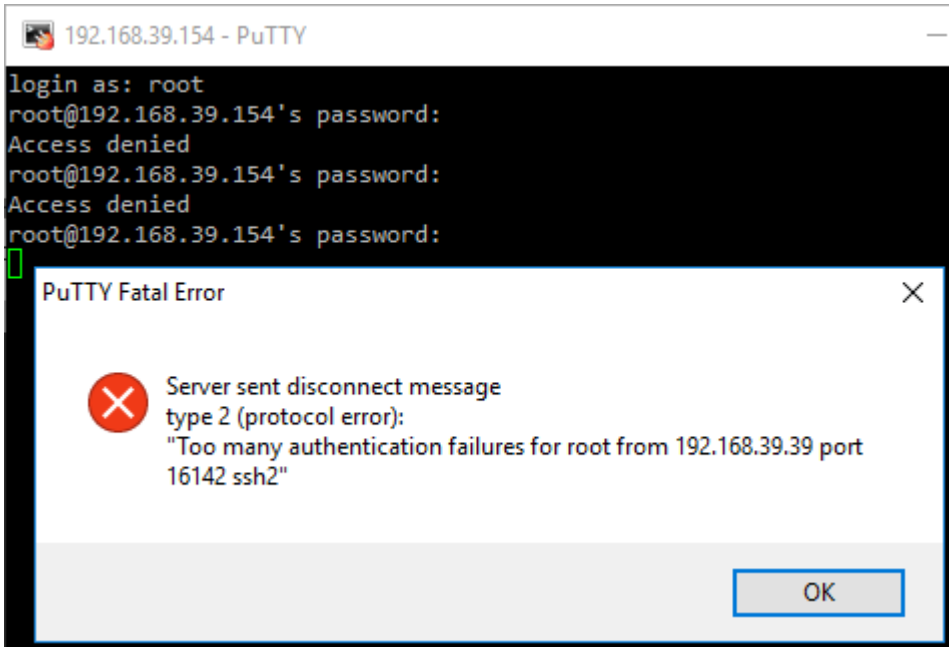
Also, add **MaxAuthTries 3** and **MaxSessions 2**. These limit the number of login attempts and concurrent logins. Then reload sshd-server with:

```
  GNU nano 2.2.6          File: /etc/ssh/sshd_config

SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin no
StrictModes yes
MaxAuthTries 3
MaxSessions 2_
```

```
systemctl restart ssh
```

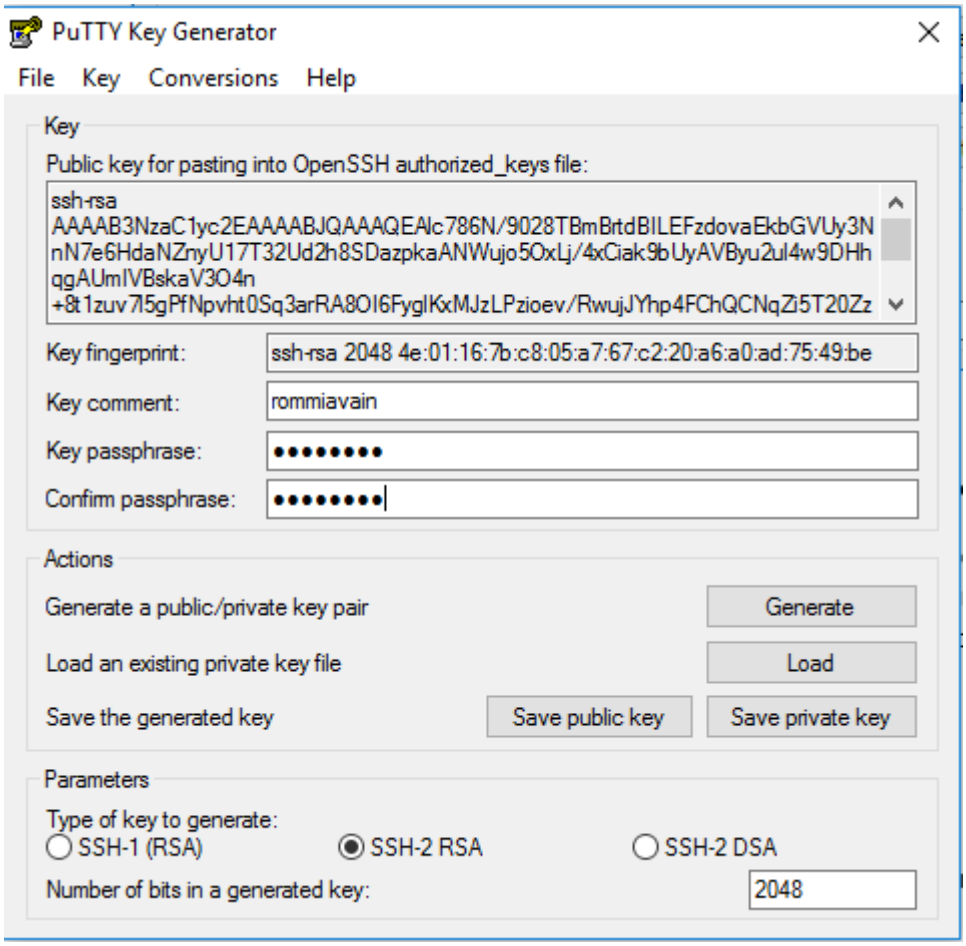
```
root@debian:~# systemctl restart ssh
```

Then try to log in to the server with PuTTY as root. You should not succeed and be disconnected after 3 tries. Login as sysadmin instead. Create a folder for SSH keys:




```
mkdir .ssh
```

Generate a SSH keypair with puttygen on the Windows machine. Use anything as comment, and ttks0800 as the passphrase. Make sure the type of key is 2048-bit RSA. Save the private key to your files and copy and paste the public key portion to:

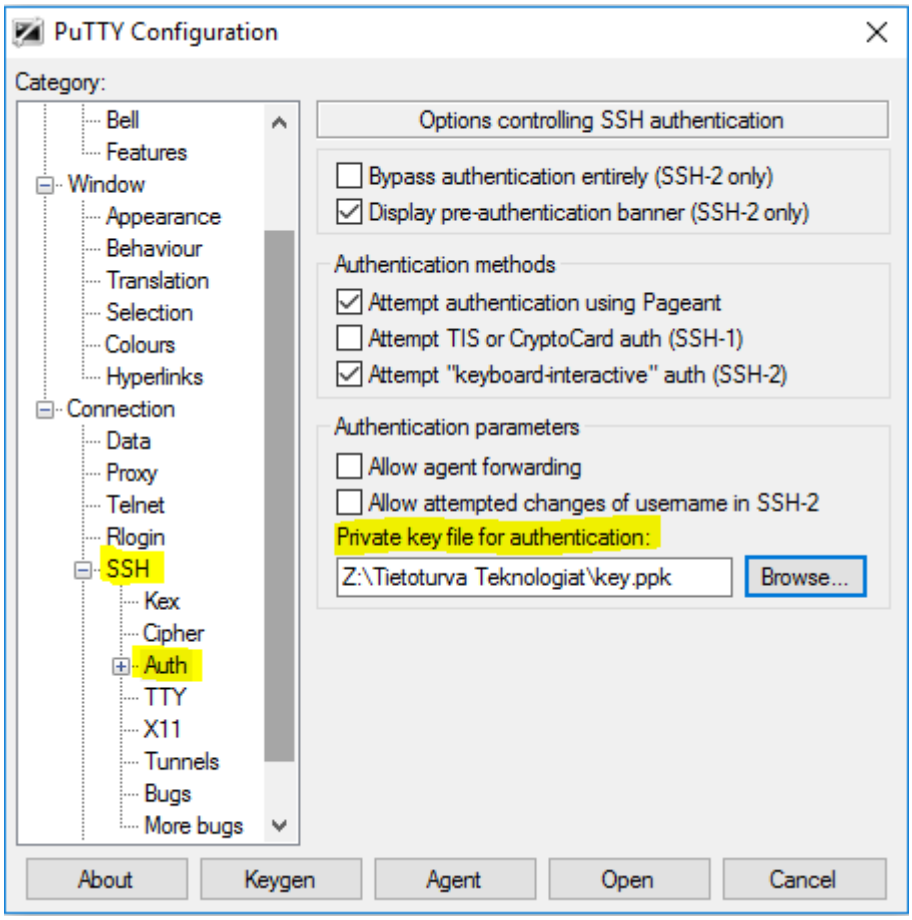


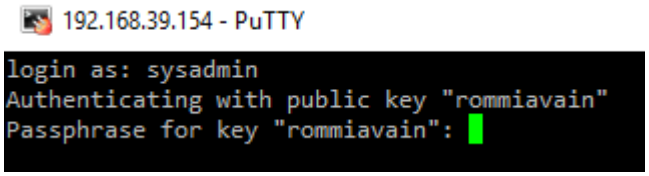
`/home/sysadmin/.ssh/authorized_keys`

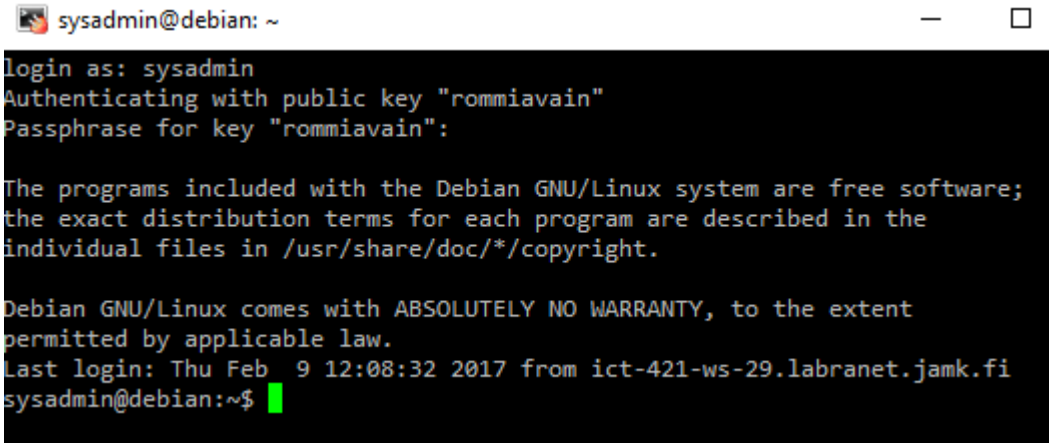
```
sysadmin@debian:~$ mkdir .ssh
sysadmin@debian:~$ sudo nano /home/sysadmin/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAlc786N/9028TBmBrtdBILEFzdovaEkbGVUy3NnN7e6HdaNZnyU17T32Ud2h8SDazpkaANWujo5OxLj/4xCiak9bUyAV$
```

Make sure the public key is copied completely and everything is on the same line.

Now you can try to login with the keypair. Open PuTTY and set **Private key file for authentication** in **Connection - SSH - Auth** to your private key file. Then open connection to the server VM. Authenticate as sysadmin, NOTE! Now you are not asked for the sysadmin password, but instead the passphrase for the keypair (ttks0800). Verify that you can login.







Explain shortly why this is better than using passwords?

- **Unused services (1p)**

This machine has no firewall, so it's a good idea to check what ports are open. Check what ports are listening with:

```
netstat -lnut
```

```
sysadmin@debian:~$ netstat -lnut
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:36388           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 ::1:631                 :::*                    LISTEN
tcp6       0      0 ::1:25                  :::*                    LISTEN
tcp6       0      0 :::50595                :::*                    LISTEN
tcp6       0      0 :::111                  :::*                    LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
udp        0      0 0.0.0.0:16878           0.0.0.0:*
udp        0      0 0.0.0.0:1016            0.0.0.0:*
udp        0      0 0.0.0.0:68              0.0.0.0:*
udp        0      0 127.0.0.1:620           0.0.0.0:*
udp        0      0 0.0.0.0:111             0.0.0.0:*
udp        0      0 0.0.0.0:41075           0.0.0.0:*
udp        0      0 0.0.0.0:631             0.0.0.0:*
udp        0      0 0.0.0.0:5353            0.0.0.0:*
udp        0      0 0.0.0.0:34657           0.0.0.0:*
udp6       0      0 :::49108                :::*
udp6       0      0 :::1016                 :::*
udp6       0      0 :::33391                :::*
udp6       0      0 :::111                  :::*
udp6       0      0 :::5353                 :::*
udp6       0      0 :::45325                :::*
```

There will be lot and most of those are not actively used for this server. Some ports are also shown multiple times for tcp/udp and IPv4/IPv6. Only services we currently need are port 22 (SSH) and port 80 (HTTP).

Start with a port number and see what process is listening to it (example shown:

```
sudo lsof -i :631 (will return cups)
```

```
sysadmin@debian:~$ sudo lsof -i :631
[sudo] password for sysadmin:
COMMAND   PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
cupsd     738 root   10u  IPv6  12126      0t0  TCP localhost:ipp (LISTEN)
cupsd     738 root   11u  IPv4  12127      0t0  TCP localhost:ipp (LISTEN)
cups-brow 739 root    5u  IPv6  12241      0t0  TCP localhost:45497->localhost:i
pp (CLOSE_WAIT)
cups-brow 739 root    7u  IPv4  12247      0t0  UDP *:ipp
```

You can also try to google for the services that use the ports shown. Then you can stop the service:

```
sudo systemctl stop cups
```

```
sysadmin@debian:~$ sudo systemctl stop cups
Warning: Stopping cups.service, but it can still be activated by:
  cups.path
  cups.socket
```

Even better, you can try to uninstall the service/software. We don't need printing, so:

```
sudo apt remove cups
```

```
sysadmin@debian:~$ sudo apt remove cups
```

Check that no seemingly vital components are on the list before proceeding.

This way, at least disable if not even remove unnecessary stuff. Search for the service names with lsof and internet. However, do NOT disable port 68 (dhclient) as this server gets IP via DHCP or port 25 (postfix/sendmail).

After you are done, you can try **sudo apt-get autoremove** which will remove all unused packages left.

```
sysadmin@debian:~$ sudo apt-get autoremove
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  avahi-daemon colord colord-data cups-browsed cups-bsd cups-client cups-core-drivers cups-daemon
  cups-filters cups-filters-core-drivers cups-ppdc cups-server-common dconf-gsettings-backend
  dconf-service foomatic-db-compressed-ppds foomatic-db-engine gir1.2-glib-2.0 hp-ppd hplip-data
  libart-2.0-2 libavahi-core7 libavahi-glib1 libcolord2 libcolorhug2 libcupscgi1 libcupsmime1
  libcupsppdc1 libdbus-glib-1-2 libdconf1 libexif12 libfile-copy-recursive-perl libfontembed1
  libgirepository-1.0-1 libgphoto2-6 libgphoto2-l10n libgphoto2-port10 libgudev-1.0-0 libgusb2
  libgutenprint2 libhpmud0 libieee1284-3 libjim0.75 libltdl7 libnss-mdns libperl5.20
  libpolkit-agent-1-0 libpolkit-backend-1-0 libpolkit-gobject-1-0 libqpdf13 libsane libsane-common
  libsane-extras libsane-extras-common libsane-hpaio libsensors4 libsnmp-base libsnmp30 libtcl8.6
  libtk8.6 libv4l-0 libv4lconvert0 libxss1 mscompress openprinting-ppds policykit-1 printer-driver-all
  printer-driver-brlaser printer-driver-c2050 printer-driver-c2esp printer-driver-cjet
  printer-driver-dymo printer-driver-escpr printer-driver-foo2zjs printer-driver-foo2zjs-common
  printer-driver-hpijs printer-driver-m2300w printer-driver-min12xxw printer-driver-pnm2ppa
  printer-driver-ptouch printer-driver-pxljr printer-driver-sag-gdi python-dbus python-dbus-dev
  python-gi python-gobject-2 python-imaging python-pexpect python-renderpm python-reportlab
  python-reportlab-accel qpdf sane-utils tcl tcl8.6 tk tk8.6 unzip update-inetd usb-modeswitch
  usb-modeswitch-data
0 upgraded, 0 newly installed, 100 to remove and 0 not upgraded.
After this operation, 96.8 MB disk space will be freed.
Do you want to continue? [Y/n] y
```

- **Firewall (1p)**

UFW (Uncomplicated Firewall) is a very user-friendly firewall for Linux. Install UFW with:

```
sudo apt install ufw
```

```
sysadmin@debian:~$ sudo apt install ufw
```

Firewall is by default off so you don't have to worry about dropping SSH connection. UFW uses "apps" so check what apps can be used:

```
sudo ufw app list
```

```
Available applications:
  AIM
  Bonjour
  CIFS
  CUPS
  DNS
  Deluge
  IMAP
  IMAPS
  IPP
  KTorrent
  Kerberos Admin
  Kerberos Full
  Kerberos KDC
  Kerberos Password
  LDAP
  LDAPS
  LPD
  MSN
  MSN SSL
  Mail submission
  NFS
  OpenSSH
  POP3
  POP3S
  PeopleNearby
  SMTP
  SSH
  Socks
  Telnet
  Transmission
  Transparent Proxy
  VNC
  WWW
  WWW Cache
  WWW Full
  WWW Secure
  XMPP
  Yahoo
  qBittorent
  svnserve
```

Find the SSH app (might be different names) and add them with:

```
sudo ufw allow "<app name>"
```

```
sysadmin@debian:~$ sudo ufw allow OPENSSH
Rules updated
Rules updated (v6)
```

Then you can turn the firewall on and see the status:

```
sudo ufw enable
```

```
sysadmin@debian:~$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
```

```
sudo ufw status verbose
```

```
sysadmin@debian:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp (OpenSSH)           ALLOW IN    Anywhere
22/tcp (OpenSSH (v6))      ALLOW IN    Anywhere (v6)
```

Check that logging is on. Then restart syslog daemon:

```
systemctl restart rsyslog
```

```
sysadmin@debian:~$ sudo systemctl restart rsyslog
```

Now UFW should log to /var/log/ufw.log. Try to browse to the server with HTTP or HTTPS and check the log for entries.

```
sysadmin@debian:~$ sudo tail /var/log/ufw.log -n 5
Feb  9 12:32:18 debian kernel: [ 3242.069954] [UFW BLOCK] IN=eth0 OUT= MAC=08:00:27:09:21:00:8c:dc:d4:46:34:a8:08:00 SRC=192.168.39.39 DST=192.168.39.154 LEN=52 TOS=0x00 PREC=0x00 TTL=1
28 ID=920 DF PROTO=TCP SPT=16487 DPT=443 WINDOW=8192 RES=0x00 SYN URGP=0
Feb  9 12:32:21 debian kernel: [ 3244.817895] [UFW BLOCK] IN=eth0 OUT= MAC=08:00:27:09:21:00:8c:dc:d4:46:34:a8:08:00 SRC=192.168.39.39 DST=192.168.39.154 LEN=52 TOS=0x00 PREC=0x00 TTL=1
28 ID=921 DF PROTO=TCP SPT=16486 DPT=443 WINDOW=8192 RES=0x00 SYN URGP=0
Feb  9 12:32:21 debian kernel: [ 3245.069714] [UFW BLOCK] IN=eth0 OUT= MAC=08:00:27:09:21:00:8c:dc:d4:46:34:a8:08:00 SRC=192.168.39.39 DST=192.168.39.154 LEN=52 TOS=0x00 PREC=0x00 TTL=1
28 ID=922 DF PROTO=TCP SPT=16487 DPT=443 WINDOW=8192 RES=0x00 SYN URGP=0
```

Lastly, add HTTP/HTTPS rules to the UFW firewall (Find the correct apps in app list).

```
sysadmin@debian:~$ sudo ufw allow http
Rule added
Rule added (v6)
sysadmin@debian:~$ sudo ufw allow https
Rule added
Rule added (v6)
```

- **Virus scanning (1p)**

Last part of the lab is to set up virus scanning. You will use ClamAV for this. Install ClamAV and update the database:

```
sudo apt install clamav clamav-daemon
```

```
sysadmin@debian:~$ sudo apt install clamav clam
av-daemon
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  clamav-base clamav-freshclam clamdscan
  libclamav7 libcurl3 libltdl7 libmspack0
```

```
sudo systemctl stop clamav-freshclam
sudo freshclam -v
```

```
sysadmin@debian:~$ sudo systemctl stop clamav-freshclam.service
sysadmin@debian:~$ sudo freshclam -v
Current working dir is /var/lib/clamav
Max retries == 5
ClamAV update process started at Thu Feb  9 12:39:26 2017
Using IPv6 aware code
Querying current.cvd.clamav.net
TTL: 364
Software version from DNS: 0.99.2
main.cvd version from DNS: 57
main.cvd is up to date (version: 57, sigs: 4218790, f-level: 60,
builder: amishhammer)
Retrieving http://db.local.clamav.net/daily.cvd
```

In order to test the AV, download an EICAR AV test file:

```
cd /home/sysadmin
wget http://www.eicar.org/download/eicar.com
```

```
sysadmin@debian:~$ wget https://www.eicar.org/download/eicar.com
--2017-02-09 12:44:02--  https://www.eicar.org/download/eicar.com
Resolving www.eicar.org (www.eicar.org)... 213.211.198.62
Connecting to www.eicar.org (www.eicar.org)|213.211.198.62|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 68 [application/octet-stream]
Saving to: 'eicar.com'

eicar.com           100%[=====================>]      68  --.-KB/s   in 0s

2017-02-09 12:44:02 (1.32 MB/s) - 'eicar.com' saved [68/68]
```

Then scan the user home directories:

```
sudo clamscan -ri /home
```

```
sysadmin@debian:~$ sudo clamscan -ri /home
/home/sysadmin/eicar.com: Eicar-Test-Signature FOUND

----------- SCAN SUMMARY -----------
Known viruses: 5737901
Engine version: 0.99.2
Scanned directories: 3
Scanned files: 6
Infected files: 1
Data scanned: 0.00 MB
Data read: 0.00 MB (ratio 0.00:1)
Time: 12.612 sec (0 m 12 s)
```

You should get a match on the eicar.com file. Note that the scanner does not remove files by default (You can do this with **--remove=yes**)

```
sysadmin@debian:~$ sudo clamscan -ri /home --remove=yes
/home/sysadmin/eicar.com: Eicar-Test-Signature FOUND
/home/sysadmin/eicar.com: Removed.
```

Lastly, turn on automatic updating of the definitions:

```
sudo systemctl start clamav-daemon
```

```
sysadmin@debian:~$ sudo systemctl start clamav-daemon
```

And add CRON job for automatic scanning nightly:

```
sudo crontab -e
```

Add following to the end:

```
00 00 * * * clamscan -i /
```

```
  GNU nano 2.2.6         File: /tmp/crontab.8PcaGV/crontab

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

0 0 * * * clamscan -i /
```