PSAwise2324Team10Aufgabe01

OS: Ubuntu Server 22.04

RAM: 2048MB

Number of CPUs: 2

Hard drive size: 7 GB

Installation of Operating System

Create the VM

The first step is creating a VM. To start with, open VirtualBox. Then, navigate to File > Import Appliance... and select /opt/psa/data/ISOs_VMs/PSA_Template.1GB.ova. Change the name of the VM in the list that pops up to a valid VM name for your team, as defined by PSA. Change the RAM setting to 2048 MB. Choose "Generate new MAC addresses...." in the drop-down menu at the bottom. After that, click Finish and the VM will be created.

Now you can start the VM. Once it has started, go to VM Devices > Optical Devices > Choose a disk file and pick the Ubuntu Server ISO image.

Install the OS

Once you reboot the VM, the OS installation will begin. You will be prompted to choose a few settings, such as language and keyboard layout. Once prompted whether you want the full installation or a minimized version, pick the option "minimized version", in order to save storage space.

When it arrives at partitioning, pick "Custom Partitioning".

First, you need to shrink the partition that already exists without corrupting any of the small amount of data that is on it. First of all, press Strg + Z to open a shell. You can use the command `lsblk` to list the current partitions. From the output, you can see that the partition that needs to be shrunk is called /dev/sda1, while the name of the entire disk itself is /dev/sda.
Before shrinking the partition, the file system on it needs to be shrunken safely. This will protect the data that's currently in the partition. Run the following commands:

```
ntfsresize /dev/sda1 -c # checks if resize would work
ntfsresize /dev/sda1 -i # shows minimum size it can be shrunk too (in out
case, 39MB. We shrunk it to 50MB to be safe.)
ntfsresize /dev/sda1 --size 50M # actually shrink it
```

If everything ran without errors, you can move onto shrinking the partition itself:

```
parted
print # prints partitions; remember "number" of the target partition (in our
case 1)
resizepart 1 50M # 1 = the target partition

quit
```

Next, you may create new partitions.

Pick a partitioning scheme. We decided to make a boot, root and home partition. Boot ~= 500MB, Root ~= 4,5GB, Home ~= 2GB and boot=/dev/sda2, root=/dev/sda3, home=/dev/sda4.

We created them as follows

- Root Partition

```
parted

mkpart primary ext4 <start>MB <end>MB # calculate start and end based on how
large you want it to be

quit

mkfs.ext4 /dev/<root>

mount /dev/<root> /
```

- Boot Partition:

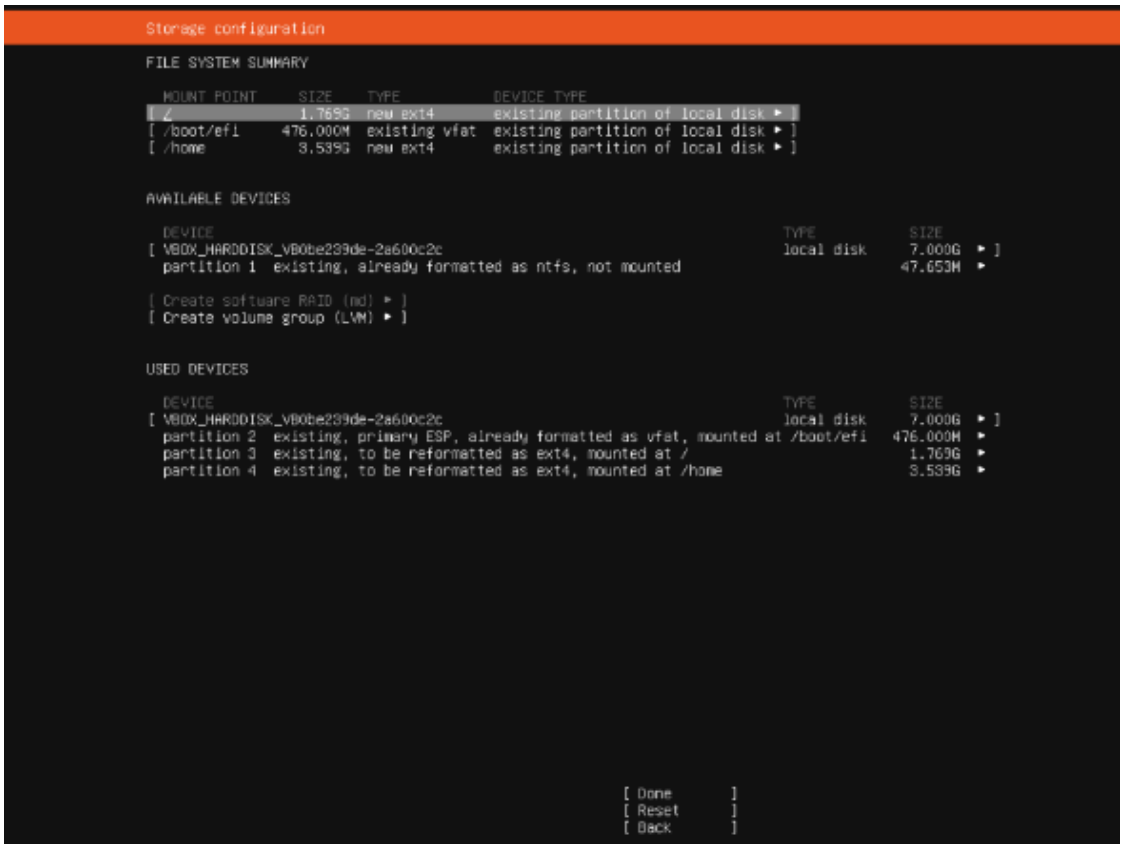
```
parted

mkpart primary fat32 <start>MB <end>MB # calculate numbers yourself
```

```
mkfs.fat /dev/<boot>

set <part_nr> boot on # use "print" to find the partition number

set <part_nr> esp on

quit

chroot /

grub-install /dev/sda
```

- Home Partition

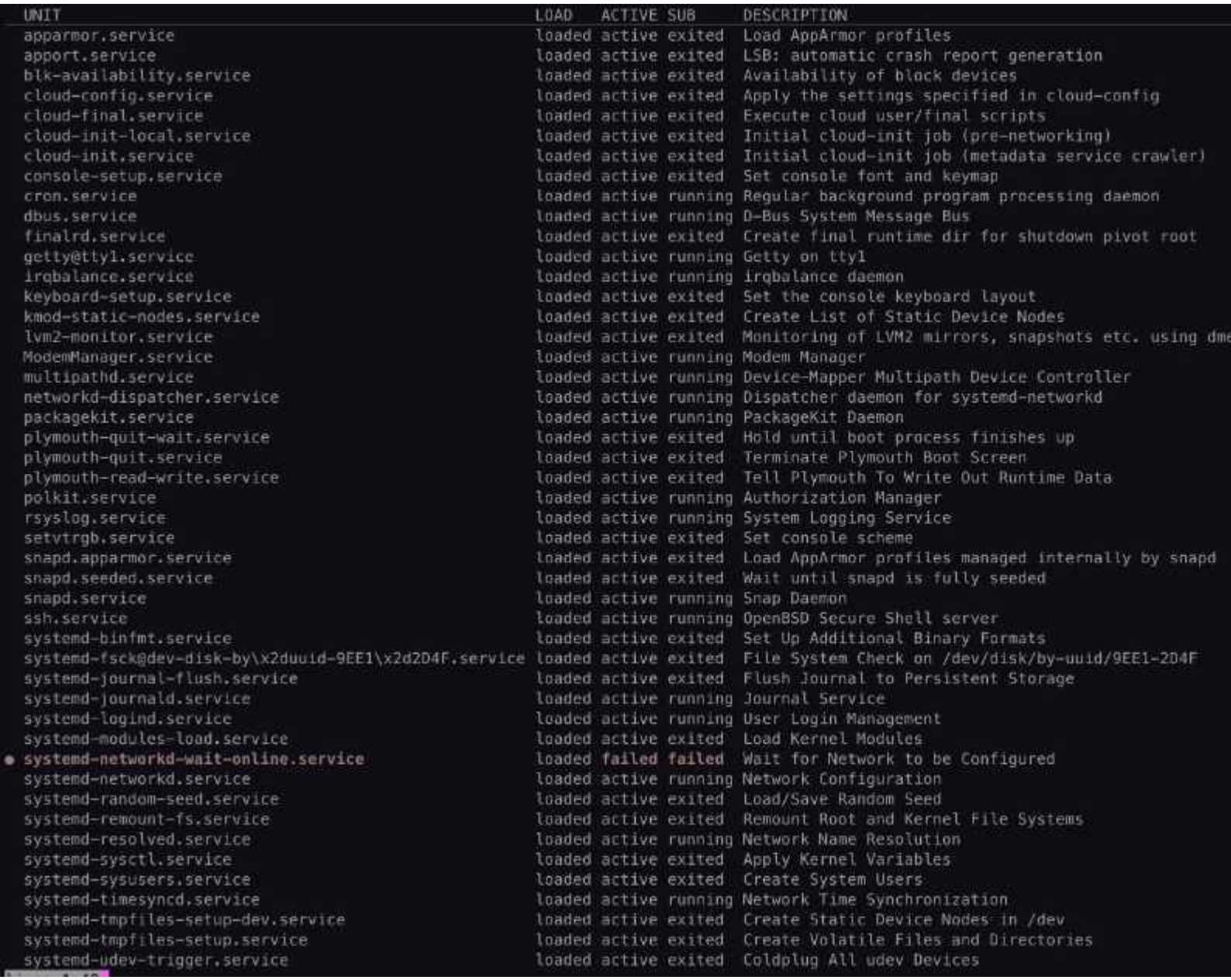
```
parted

mkpart primary ext4 <start>MB <end>MB # calculate numbers yourself

quit
```

After this, type `exit` to close the shell and return to the installation menu. Click "Reset" and the changes should be applied. Now you can pick a mount for the home partition. On the hard disk, choose "use this device as boot disk". In some cases, we also had to delete and create root again from scratch, because the installation menu didn't carry the changes over properly. Once everything looks as it should, click "Done" and the installation should run to completion.
It should look something like this (in this example, root is still a bit too small)



Services

```
sudo systemctl list-units --type==service
```



```
  UNIT                                                      LOAD   ACTIVE SUB     DESCRIPTION
  apparmor.service                                          loaded active exited  Load AppArmor profiles
  apport.service                                            loaded active exited  LSB: automatic crash report generation
  blk-availability.service                                  loaded active exited  Availability of block devices
  cloud-config.service                                      loaded active exited  Apply the settings specified in cloud-config
  cloud-final.service                                       loaded active exited  Execute cloud user/final scripts
  cloud-init-local.service                                  loaded active exited  Initial cloud-init job (pre-networking)
  cloud-init.service                                        loaded active exited  Initial cloud-init job (metadata service crawler)
  console-setup.service                                     loaded active exited  Set console font and keymap
  cron.service                                              loaded active running Regular background program processing daemon
  dbus.service                                              loaded active running D-Bus System Message Bus
  finalrd.service                                           loaded active exited  Create final runtime dir for shutdown pivot root
  getty@tty1.service                                        loaded active running Getty on tty1
  irqbalance.service                                        loaded active running irqbalance daemon
  keyboard-setup.service                                    loaded active exited  Set the console keyboard layout
  kmod-static-nodes.service                                 loaded active exited  Create List of Static Device Nodes
  lvm2-monitor.service                                      loaded active exited  Monitoring of LVM2 mirrors, snapshots etc. using dme
  ModemManager.service                                      loaded active running Modem Manager
  multipathd.service                                        loaded active running Device-Mapper Multipath Device Controller
  networkd-dispatcher.service                               loaded active running Dispatcher daemon for systemd-networkd
  packagekit.service                                        loaded active running PackageKit Daemon
  plymouth-quit-wait.service                                loaded active exited  Hold until boot process finishes up
  plymouth-quit.service                                     loaded active exited  Terminate Plymouth Boot Screen
  plymouth-read-write.service                               loaded active exited  Tell Plymouth To Write Out Runtime Data
  polkit.service                                            loaded active running Authorization Manager
  rsyslog.service                                           loaded active running System Logging Service
  setvtrgb.service                                          loaded active exited  Set console scheme
  snapd.apparmor.service                                    loaded active exited  Load AppArmor profiles managed internally by snapd
  snapd.seeded.service                                      loaded active exited  Wait until snapd is fully seeded
  snapd.service                                             loaded active running Snap Daemon
  ssh.service                                               loaded active running OpenBSD Secure Shell server
  systemd-binfmt.service                                    loaded active exited  Set Up Additional Binary Formats
  systemd-fsck@dev-disk-by\x2duuid-9EE1\x2d2D4F.service     loaded active exited  File System Check on /dev/disk/by-uuid/9EE1-2D4F
  systemd-journal-flush.service                             loaded active exited  Flush Journal to Persistent Storage
  systemd-journald.service                                  loaded active running Journal Service
  systemd-logind.service                                    loaded active running User Login Management
  systemd-modules-load.service                              loaded active exited  Load Kernel Modules
● systemd-networkd-wait-online.service                      loaded failed failed  Wait for Network to be Configured
  systemd-networkd.service                                  loaded active running Network Configuration
  systemd-random-seed.service                               loaded active exited  Load/Save Random Seed
  systemd-remount-fs.service                                loaded active exited  Remount Root and Kernel File Systems
  systemd-resolved.service                                  loaded active running Network Name Resolution
  systemd-sysctl.service                                    loaded active exited  Apply Kernel Variables
  systemd-sysusers.service                                  loaded active exited  Create System Users
  systemd-timesyncd.service                                 loaded active running Network Time Synchronization
  systemd-tmpfiles-setup-dev.service                        loaded active exited  Create Static Device Nodes in /dev
  systemd-tmpfiles-setup.service                            loaded active exited  Create Volatile Files and Directories
  systemd-udev-trigger.service                              loaded active exited  Coldplug All udev Devices
```

ssh	allows secure remote login
apparmor	mandatory access control system. Permits programs access to specific resources only.
apport	reports and collects information about system crashes and other issues
blk-availability	indicates whether block devices (e.g., hard drive) are available
cloud-init und andere cloud Dienste	generates, for example, hostname, ssh keys, and performs other organizational tasks in this area
console-setup, keyboard-setup	basic setup of font and keyboard in the console
cron	job scheduler - we don't plan to actively use this service but have kept it in case any program/process requires it
dbus	D-Bus Message bus
finalrd	creates final runtime directory
getty@tty1	manages terminal
irqbalance	distributes hardware interrupts across processors to improve performance
k-mod-static-nodes	generates a static device node list
lvm2-monitor	manages LVM2 mirror, etc.
modemManager	manages modems, but we don't need it - **removed**
multipathd	identifies failed paths if multiple paths lead to a device

networkd-dispatcher	generates dispatches for network management
packagekit	makes installing and updating software easier
plymouth	responsible for the graphical boot screen
polkit	manages communication from unprivileged processes to privileged processes
rsyslog	system logging
setvtrgb	sets console scheme
snapd	manages snaps
systemd	various init processes at system startup

Since we chose a minimized installation, we don't have a lot of superfluous services. The one we did identify, we uninstalled using `sudo apt purge modemmanager`.

Root Access

Run `sudo wget “https://www.net.in.tum.de/teaching/ws2324/psa/ssh_key.pub” - O /root/home/.ssh/authorized_keys`. Then, make sure that “PasswordAuthentication no” and “PermitRootLogin prohibit-password” are set in /etc/ssh/sshd_config.

For remote access in general, you also need to configure port forwarding as follows:

Open VirtualBox and go to Settings > Network > Adapter 1 > Advanced > Port forwarding, then set a port forwarding like this (you can find rules for appropriate port numbers in the PSA wiki)

Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
ssh	TCP	0.0.0.0	61002		22

Users

To make the creation of new users easier, we created a Python script that handles the task. For this, we used BeautifulSoup to extract the public keys of all participants from the wiki page and stored them in a Pandas DataFrame.

In the function text_to_df(), the data extracted from the website is formatted for the DataFrame, and a column indicating the team member's number is added. This script makes the assumption that there will only be teams of two.

The add_users() function implements the main part of the task: It iterates over the rows of the DataFrame and checks whether the that user already exists. If not, using the Subprocess packages, the required Bash commands are executed to create the user, using their UID, primary GID, and secondary GID. The first time the user logs into our server, they will be prompted to create a new password.

```
#!/usr/bin/python3
import subprocess
import requests
from bs4 import BeautifulSoup as bs
import pandas as pd
import os
from getpass import getpass

URL =
'https://psa.in.tum.de/xwiki/bin/view/PSA%20WiSe%202023%20%202024/Public%20Key
s/'


def check_group_exists(name):
    try:
        subprocess.run(['getent', 'group', name], check=True,
capture_output=True)
        return True
    except subprocess.CalledProcessError:
        return False


def check_user_exists(name):
    try:
        subprocess.run(['getent', 'passwd', name], check=True,
capture_output=True)
        return True
    except subprocess.CalledProcessError:
        return False


def add_team_user_numbers_col(df):
    mem_numbers = []
    member_nr_bool = df['team'].diff().eq(0)
    for x in member_nr_bool:
        if not x:
            mem_numbers.append(1)
        else:
            mem_numbers.append(2)
    df_res = df.assign(member_nr=mem_numbers)
    return df_res


def text_to_df(text):
    tokenized = text.split("|")
    temp_tokenized = tokenized
```

```
    del temp_tokenized[:6]
    new_list = [temp_tokenized[x:x + 5] for x in range(0, len(temp_tokenized)
- 2, 5)]
    list_with_nulls = ['null' if s == '\xa0' else s for s in new_list]
    df = pd.DataFrame(list_with_nulls, columns=['username', 'gecos', 'team',
'pub_key', 'lrz_id'])
    df['team'] = [x.removeprefix("Team ") for x in df['team']]
    df['team'] = [int(x) for x in df['team']]
    df = add_team_user_numbers_col(df)
    return df


def add_authorized_keys(name, pub_key):
    home_directory = f'/home/{name}/'
    authorized_keys_path = home_directory + ".ssh/authorized_keys"
    with open(authorized_keys_path, "a") as auth_keys_file:
        auth_keys_file.write(pub_key + "\n")


def change_user_password(user, new_password):
    passwd_process = subprocess.Popen(['sudo', 'passwd', user],
stdin=subprocess.PIPE, stdout=subprocess.PIPE,
                                      stderr=subprocess.PIPE,
universal_newlines=True)
    passwd_process.communicate(input=f"{new_password}\n{new_password}\n")
    return passwd_process.returncode


def add_users(user_df):
    temp_df = user_df[['username', 'team', 'member_nr', 'pub_key']]
    # iterate over df and check if user exists
    for index, row in temp_df.iterrows():
        if not check_user_exists(row['username']):
            # calculate uid
            uid = 1000 + 100 * int(row['team']) + int(row['member_nr'])
            # calculate secondary gid
            gid = 1000 + 100 * int(row['team'])
            # create secondary group
            if not check_group_exists(f'psa2324team{row["team"]}'):
                subprocess.run(['sudo', 'groupadd', '-g', str(gid),
f'psa2324team{row["team"]}'])
            # create primary group
            subprocess.run(['sudo', 'groupadd', '-g', str(uid),
row['username']])
            # create user with primary gid = uid
            subprocess.run(
                ['sudo', 'useradd', '-m', '-d', f'/home/{row["username"]}', '-
s', '/bin/bash', '-u', str(uid), '-g', str(uid), '-G',
                 str(gid), row['username']],
                check=True)
            # set password to "psa", users can change it on their own later on
            rc = change_user_password(row['username'], 'psa')
            if rc == 0:
                print(f"Password for user {row['username']} changed
successfully.")
            else:
```

```
                print(f"Failed to change password for user {row['username']}.
Return code: {rc}")
            # create ssh directory
            filepath = f'/home/{row["username"]}'
            if not os.path.exists(filepath):
                os.mkdir(filepath)
            if not os.path.exists(f'{filepath}/.ssh'):
                os.mkdir(f'{filepath}/.ssh')
            os.chmod(f'{filepath}/.ssh', 0o700)
            os.chown(filepath, uid=uid, gid=gid)
            # add user key
            add_authorized_keys(row['username'], row['pub_key'])


if __name__ == "__main__":
    # check if psa group already exists, if not create new group
    if not check_group_exists('psa2324'):
        subprocess.run(['sudo', 'groupadd', 'psa2324', '-g', '1099'],
text=True)

    username = input("Input email: ")
    password = getpass("Input password: ")

    # parse user list from website
    s = requests.session()
    response = s.post(URL, auth=(username, password))
    index_page = s.get(URL)
    soup = bs(index_page.text, 'html.parser')
    results = soup.find(id="xwikicontent")
    temp_text = results.get_text("|")

    ######################
    # TODO temporary bug fix weil falscher input in der liste
    temp_text = temp_text.replace(
        '|ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIFbetiUdtIMVZ+x0VR0PCdL+IOhcVu5CW++xbEIJZTGd domi-
fresh@domifresh-hplaptop15dw1xxx|',
        '')
    ######################

    # format user data
    users = text_to_df(text=temp_text)
    # for every entry in user list create a new user if not already existing
and add to group psa
    add_users(user_df=users)
```

The script must be manually executed as needed. In the future, one could consider running the script using a systemd service that becomes active, for example, every 24 hours. However, at the moment we have decided against it because accessing the user list requires a login containing sensitive data, and we don't want to weaken our security. In a context where we would need to administer a larger number of servers, we would likely opt for a tool like Puppet to manage the users on our servers.