Lab - Linux BASH Shell Scripting – Task Scheduling

Introduction to Crontab

In windows, we use the AT command to schedule a job, in Linux, we use Crontab. Crontab is a list of commands which enables us to schedule a repetitive task to run on a schedule. This is also the name of the command used to manage the list.

Cron is the system process which automatically performs tasks per a set schedule. The schedule is called the crontab. Crontab is the name of the program used to edit the schedule.

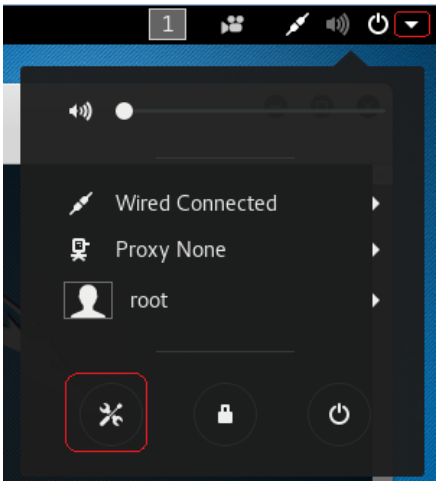
We have two parts being used, Cron which is the system process used to schedule tasks and Crontab which is used to edit the schedule.

Begin the lab…

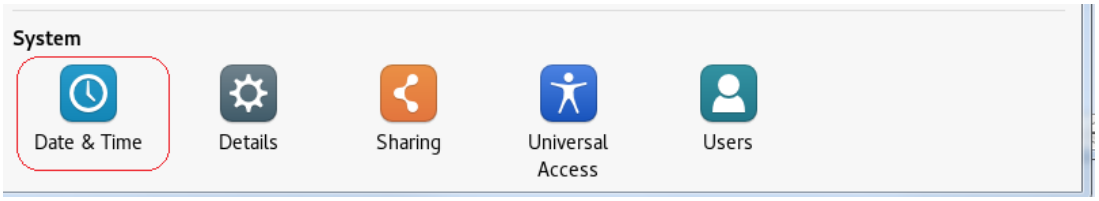
Start your Kali virtual machine. **Ensure your clock in Kali is set to the correct time!**

Sync your clock in Kali:

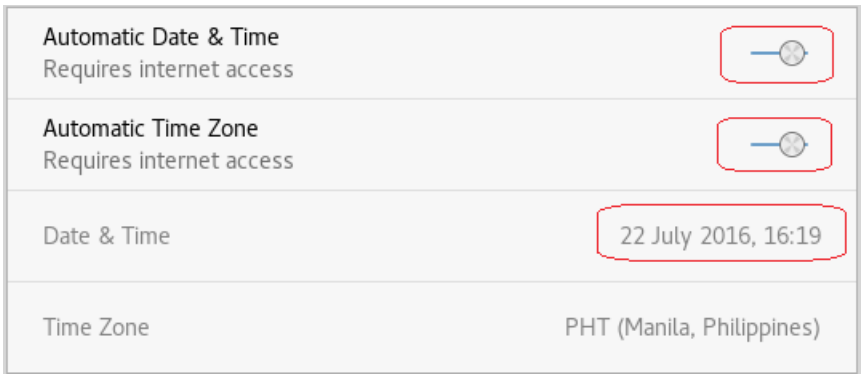
Click on the down arrow in the upper right corner and then go to system tools:



Go down to the section marked System and click on Date & Time.



On the next screen push the buttons all the way to the right for automatic sync.

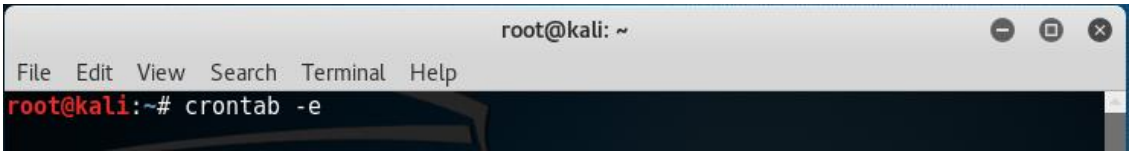


Close the systems tools and return to your Kali desktop.

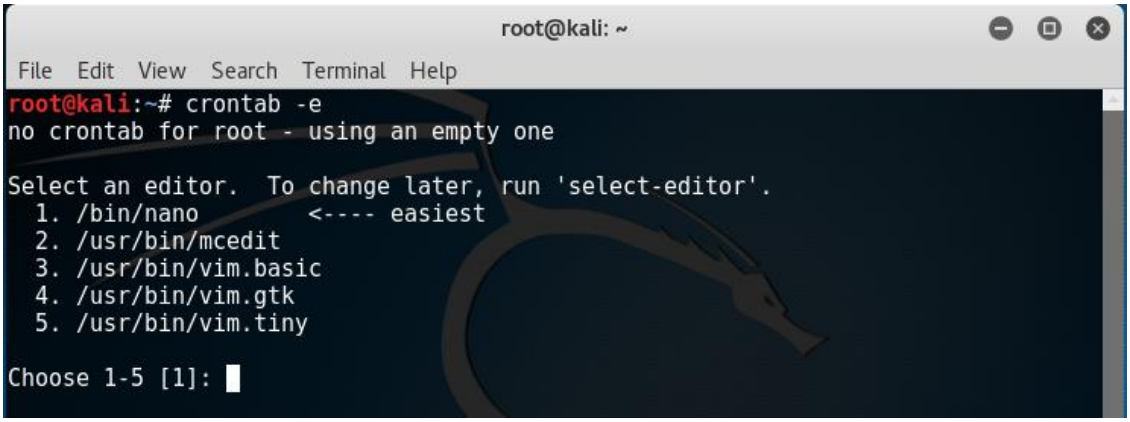
Open a terminal prompt.

And let's begin….

At the terminal prompt, type **crontab -e** for edit.



If prompted to select your choice of editor, choose option 1 or just hit enter as option 1 is the default choice.



Crontab opens using the nano file editor.

Scroll down to where it says, "For example…. Here we find an example of a scheduled backup job. Notice the job is remarked out using the # sign. Anything following the # sign will be ignored and treated as just a remark or comment. If I wanted the example backup job to run, I could just remove the # sign and save the file as a script, and it would happen. The zero (0) in the command represents the minutes, the 5 represent the hour of the day using military or 24-hour clock, the first * represent the day of the month (* = any day of the month), the second *

represents the month (* = Any month) and the numeral 1 represents the day of the Monday is represented using the numeral 1.

Us the following chart to help better understand how this works:

```
┌──────────── minute (0 - 59)
│ ┌────────── hour (0 - 23)
│ │ ┌──────── day of month (1 - 31)
│ │ │ ┌────── month (1 - 12)
│ │ │ │ ┌──── day of week (0 - 6 => Sunday - Saturday, or
│ │ │ │ │                 1 - 7 => Monday - Sunday)
↓ ↓ ↓ ↓ ↓
* * * * * command to be executed
```

It is possible to replace numbers with shortened name of days, such as MON, THU, etc.

Finally, if you want to specify day by day, you can separate days with commas, for example, SUN, MON, THU will execute the command only on Sundays, Mondays on Thursdays.

Use your down arrow to see the syntax of how to format a schedules task.

We are now ready to schedule our first task.

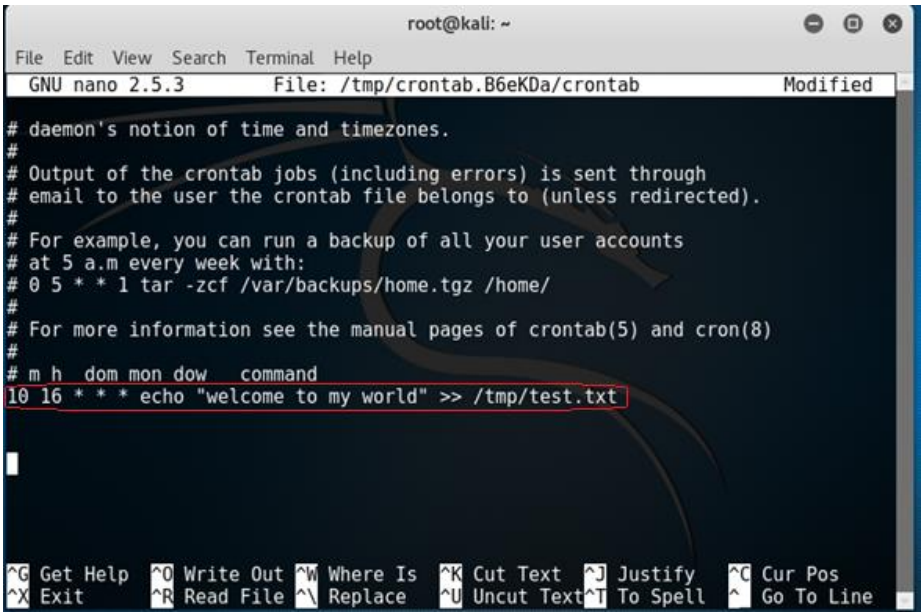
We will schedule a job to run within the next five minutes or so. This is so we can see the job launch.

Look at your clock. Now add five minutes to the time and insert this into the next available line of the Crontab scheduler, using military time, add the current hour and follow that up with three *'s, with a single space between each.

Use the echo command and have something outputted to the screen when the job does run.

Tell Crontab to save the output to a file in the temp directory.

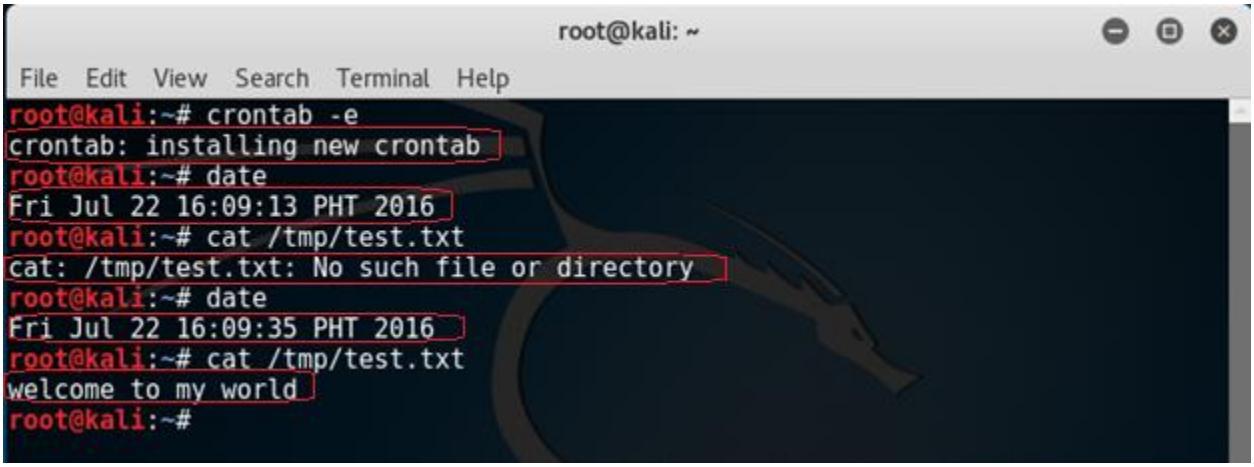
Here's my job:



Let's break down my job.

At 1610 hrs., any day of the month, any month of the year and any day of the week, this job will output "welcome to my world" and save the output to a file in the tmp directory named test.txt.

Type in CRT+X to save the job hit Y for yes and then hit enter. Here's the confirmation the job saved correctly:



At the next prompt, type the **date** command for the current day and time.cat

The job has not run yet so when I try and read the file using the **cat** command; it says no such file name or directory. The file will not be created until the job runs.

Once the clock hit 1610 hrs, I run the **cat** command again, and now I see the file was generated with the echo command and the variable of "welcome to my world" and saved to my test.txt file located in the tmp directory.

Summary –

You can use the Crontab to schedule just about any task you can think of. No need to reinvent the wheel when it comes to BASH scripting and scheduling jobs. Use the Internet to find the BASH script or the Crontab command you're looking for.

End of the lab!