blnk

Secure Backup/Restore

Overview

The goal of this task is to write 2 bash scripts that perform secure encrypted backup and restore functionality. You should be able to maneuver through the Linux configuration files and be able to schedule running the backup script on predefined times. Finally, you will need to copy the backup to a remote server.

Specifications

You are required to write a bash script that backup the content of a specific directory. The backup script, that you should call **backup.sh**, should backup only the directories within the target given specific directory, each directory in a separate compressed tar file. All the files within the directory should be backed up in a separate compressed tar file. After the backup is done, it should be copied to a remote server. Consequently, the backup script should receive from the user 4 input command-line parameters; the first parameter is the directory to be backed up, the second parameter is the directory which should store eventually the backup, and the third parameter is an encryption key that you should use to encrypt your backup and the fourth parameter is number of days (n) that the script should use to backup only the changed files during the last n days.

The restore script, which you should call **restore.sh**, should work in a reverse way and should be able to restore a backup that was originally taken by **backup.sh**. Consequently, the restore script should receive from the user precisely 3 command-line parameters; the first parameter is the directory that contains the backup, the second parameter is the directory that the backup should be restored to, and the third parameter is the decryption key that should be used to restore the backup.

Finally, you should amend the cron configuration files such that the backup script is executed every day. You should read the cron and the crontab man pages and read about the cron and research it well to be able to configure it. You will also need to carry out all exercise tasks using the command-line.

Milestones

Backup Script (backup.sh):

1. You should check that the script received 4 parameters from the user and validate the parameters. You should report errors if any before halting or aborting the script. e.g. user did not pass enough parameters, provided directories are not valid ones, etc.

2. Your program should print a help message indicating how it should be used if it is invoked without any parameters.

3. You should store all the parameters passed via the command line into bash variables.

4. You take a snapshot of the full date and store it in a bash variable to be used later. You should replace any white space or colon with an underscore as we will use this variable to create directory names and file names; you can use sed to do that :)

5. You need to create a directory whose name should be equivalent to the date taken in point #4, under the backup directory provided as the second command-line parameter.

6. Your script should loop over all directories under the backup directory provided as the first user command-line parameter and check for the modification date to backup only modified files within the number of days specified by the fourth parameter.

7. You should create a tar.gz file using the tar command and the necessary switches under the created new backup directory. The filename should be **<original directory name>_<date>.tgz**. The "date" is the date acquired in point #4.

8. Within the loop, use the gnupg tool to encrypt the file created in point #7, using the provided key on the command line, in a new file with the same name followed by ".gpg".

9. Very important: you need to delete the original tar file and keep the encrypted one.

10. After you are done with all the directories, you should enumerate all the files located directly under the backup main directory and group them into one tar.gz file and encrypt it in the same way.

- It is highly recommended to add the files into the tar archive one by one through using the tar update switch.
- For the first file you need to use the create switch.
- Then at the end compress the tar file using gzip and delete the tar file.
- Encrypt the tar.gz file using gnupg tool and delete the tar.gz file.

11. After the backup is done, you should copy the backup into a remote server using scp.

Restore Script (restore.sh):

1. The restore script works the same way as the backup but in the reverse order.

2. Create a temp directory under the restore directory; command-line parameter #2.

3. Loop over all the files in the backup directory; command-line parameter #1.

4. Use gnupg tool to decrypt the files one by one and store the resulting file under the temp directory created in point #2. Use the decryption key provided via the command-line user input #3.

5. Loop over the files stored in the temp directory and extract them one by one under the restore directory; command-line parameter #2.

Important Notes:

1. You are required to perform all possible validations needed and your scripts should react to them accordingly.
2. You should modularize your script functionalities into bash functions:
    - Create a separate file, and call it **backup_restore_lib.sh** that includes 4 functions: **validate_backup_params**, **backup**, **validate_restore_params**, **restore**.

- The two scripts **backup.sh** and **restore.sh** should source **backup_restore_lib.sh** and invoke the corresponding functions.

3. You should state all your assumptions in your design document.

What to submit

1. Use tar to build an archive of all your code.
2. A design document that includes all your design decisions and assumptions.
3. A small readme file explaining how to use your backup tool
4. Compress all your work: source code, design document, readme file, and any extra information into a tgz archive. You should name your archive in the specific format <Name>_Devops_Task.tgz. Finally, send it via email by replying to the same mail thread of the task.