Secure Backup/Restore Documentation

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 backs up 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.

backup_restore_lib.sh

This is a Bash script that provides functionality for backing up and restoring files. The script includes two main functions named `backup` and `restore`, along with two helper functions named `validate_backup_params` and `validate_restore_params`.

The `validate_backup_params` function takes four parameters: TARGET_DIR, DESTINATION_DIR, ENCRYPTION_KEY, and DAYS. The function sets these parameters as environment variables, and then checks if the script was called with the correct number of parameters. If the correct number of parameters was not passed, or if any of the parameters are invalid, the function prints an error message and returns an error code.

The `backup` function sets some variables for the remote server, creates a backup of the files in the `TARGET_DIR` directory and saves them in the `DESTINATION_DIR` directory. The backup files are encrypted using the `ENCRYPTION_KEY`. The `DAYS` parameter is used to backup only the changed files during the last n days. The backup files are compressed and encrypted using the `gpg` command. The compressed backup file is then transferred to a remote server using the `scp` command.

The `validate_restore_params` function takes three parameters: `RESTORE_DIR`, `DESTINATION_RESTORE_DIR`, and `DECRYPTION_KEY`. These parameters are used to validate the restore process parameters before starting the restore. The function checks if the parameters are valid, and if not, it displays an error message and returns a non-zero exit code.

The `restore` function restores the files that were backed up using the `backup` function. The backup files are decrypted and decompressed using the `gpg` and `tar` commands, respectively. The files are then restored to the `DESTINATION_RESTORE_DIR` directory.

backup.sh

This is a Bash script that includes a call to the "backup_restore_lib.sh" script and two functions, "validate_backup_params" and "backup". The script is designed to perform a backup operation with user-defined parameters.

The script first calls the "validate_backup_params" function, which takes four parameters: $1 (TARGET_DIR), $2 (DESTINATION_DIR), $3 (ENCRYPTION_KEY), and $4 (DAYS). This function validates the parameters and returns an exit code, which is stored in the "validate_exit_code" variable.

If the exit code is equal to 1, the script prints an error message indicating that the backup failed. If the exit code is not equal to 1 or 2, the script calls the "backup" function.

The "backup" function creates a backup of the files in the "TARGET_DIR" directory and saves them in the "DESTINATION_DIR" directory. The backup files are encrypted using the "ENCRYPTION_KEY" and compressed using the "gzip" command. The compressed backup file is then transferred to a remote server using the "scp" command.

restore.sh

This is a Bash script that includes a call to the "backup_restore_lib.sh" script and two functions, "validate_restore_params" and "restore". The script is designed to perform a restore operation with user-defined parameters.

The script first calls the "validate_restore_params" function, which takes three parameters: $1 (RESTORE_DIR), $2 (DESTINATION_RESTORE_DIR), and $3 (DECRYPTION_KEY). This function validates the parameters and returns an exit code, which is stored in the "validate_exit_code" variable.

If the exit code is equal to 1, the script prints an error message indicating that the restore failed. If the exit code is not equal to 1 or 2, the script calls the "restore" function.

The "restore" function restores the files that were backed up using the "backup" function. The backup files are decrypted and decompressed using the "gpg" and "tar" commands, respectively. The files are then restored to the "DESTINATION_RESTORE_DIR" directory.

Cron Jobs

To schedule a daily cron job, you can use the crontab -e command to edit the crontab file. The general format of a cron job is * * * * * command, where each asterisk represents a time unit from left to right: minute, hour, day of the month, month, and day of the week, respectively.

For example, to run a backup script every day at midnight, you can add the following line to the crontab file: 0 0 * * * /home/user/backup.sh. This will execute the backup.sh script located in the /home/user directory at 12:00am every day.

Note that you may need to set environment variables in the script or add them as hard-coded values for the cron job to run correctly. It's also important to thoroughly test your cron job to ensure that it executes as expected and to monitor it periodically for any issues.

Overall

This Tool provides a simple way to perform encrypted backups and restores of directories, while checking for errors and validating input parameters. You can customize the script by changing the values of the variables for the remote server or modifying the validation and error messages to suit your needs.