

Lindsay Graham

Scripting for Security Part 2

Scripting

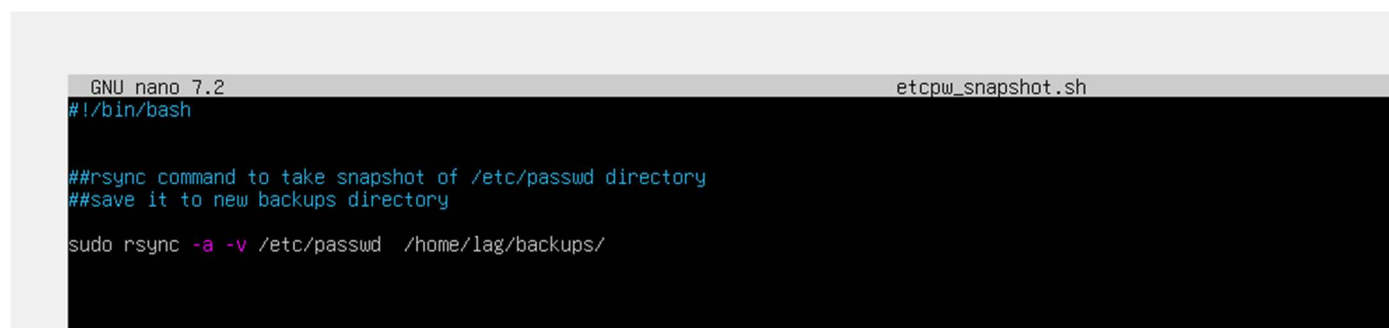
Link to scripts: <https://github.com/lag249/Linux.git>

Ubuntu Server

Example 1. Take a snapshot of users every other hour (Use a cron job for this) to see if there is any suspicious adding/removing of users

Rsync can be a useful tool for creating backups or taking snapshots of files or directories (Rubel, 2002). I chose to use rsync to monitor the `/etc/passwd` file to monitor users. First, I wrote a basic script that would copy the `/etc/passwd` file and save it to the backups directory. The `-a` option after rsync specifies archive mode, which copies files recursively and preserves permissions, timestamps, etc. (Sheldon, 2023). The `-v` option specifies that rsync should be used in verbose mode so that there is detailed information on the process.

Figure 1.1 Rsync script to take a snapshot of `/etc/passwd` directory



```
GNU nano 7.2 etcpw_snapshot.sh
#!/bin/bash

##rsync command to take snapshot of /etc/passwd directory
##save it to new backups directory

sudo rsync -a -v /etc/passwd /home/lag/backups/
```

The syntax to use rsync in this script uses the command **`sudo rsync -a -v /etc/passwd(source file) /home/lag/backups/(destination)`**.

Next, I set up a cron job to run the snapshot script every other hour and also to create an output file, `etcpw_log.txt`. The five asterisks at the beginning of any cron job signify the minute, hour, day of month, month, and day of week (StackOverflow, 2015). To set up the cron job, type **`crontab -e`** to go to the configuration file. The syntax to set the time portion of the job to run every other hour is **`* * * * *`** `*/2 * * *` (Wilkinson, 2024). Then, specify the script to run, and an output file if that is desired.

Figure 1.2 Image of cron job to run rsync script

```
* * * * * rsync -a /home/lag/scripts/etcpw_snapshot.sh >> /home/lag/backups/etcpw_log.txt

crontab: installing new crontab
```

To confirm that everything ran as expected, navigate to the backups directory and use **ls** to see the contents. Figure 1.3 shows the snapshot of **/etc/passwd** and the log file.

Figure 1.3 Contents of backup directory after cron job

```
lag@ubuntu:~/backups$ ls
etcpw_log.txt passwd
lag@ubuntu:~/backups$
```

To view the contents of the files, you can open them using a text editor like nano.

Figure 1.4 The contents of the passwd snapshot

```
GNU nano 7.2                                passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailng List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534:/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
systemd-timesync:x:997:997:systemd Time Synchronization:/:/usr/sbin/nologin
dhcpcd:x:100:65534:DHCP Client Daemon,,,:/usr/lib/dhcpcd/bin/false
messagebus:x:101:102:/nonexistent:/usr/sbin/nologin
systemd-resolve:x:992:992:systemd Resolver:/:/usr/sbin/nologin
pollinate:x:102:1:/var/cache/pollinate:/bin/false
polkitd:x:991:991:User for polkitd:/:/usr/sbin/nologin
syslog:x:103:104:/nonexistent:/usr/sbin/nologin
uuid:x:104:105:/run/uuid:/usr/sbin/nologin
tcpdump:x:105:107:/nonexistent:/usr/sbin/nologin
ss:x:106:106:TPM software stack,,,:/var/lib/tpm/bin/false
landscape:x:107:108:/var/lib/landscape:/usr/sbin/nologin
fwupd-refresh:x:989:989:Firmware update daemon:/var/lib/fwupd:/usr/sbin/nologin
usbmux:x:108:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
sshd:x:109:65534:/run/ssh:/usr/sbin/nologin
lag:x:1000:1000:lindsay:/home/lag:/bin/bash
postfix:x:110:111:/var/spool/postfix:/usr/sbin/nologin
hattie:x:1001:1001:/home/hattie:/bin/sh

[File 'passwd' is unwritable]
```

Figure 1. 5 Contents of etcpw_log.txt

```

GNU nano 7.2                                     etcpw_log.txt
-rwxrwxrwx      157 2024/11/11 13:21:34 etcpw_snapshot.sh
-rwxrwxrwx      157 2024/11/11 13:21:34 etcpw_snapshot.sh

```

Note: I am unsure what the output in the log file signifies. I can't find a source that explains it, but I assume it is keeping track of when the script was run. Please note that for testing, I used all asterisks for the cron job before setting it to every other hour. That is why the time in the log does not reflect that the script has been run every other hour.

Example 2. Write a script to back up your most important information and your logs

The three directories that I thought were most important to back up were the /etc, /root, /home/scripts, and /var directories. The /etc directory is important to back up because it contains all of the server's system configuration files (Hess, 2020). This includes information related to application configurations, users, passwords, and start up files. If a full restore of the server was needed, the /etc directory would be crucial for bring the system back to an operational state. The /home/scripts directory has important scripts which I as a user would not want to lose. The /var directory is important because it contains all of the logs, crontabs, and any databases that may be on a system. Lastly, the root directory would be important to backup because it holds downloads, configurations, and scripts that are unique to the system. The -z option in the backup script enables file compression when the files are being transferred (Sheldon, 2023). The snapshots are saved as compressed gzip files to save space.

Example 2.1 Script to back up /etc, /home/scripts, /root, and /var directories

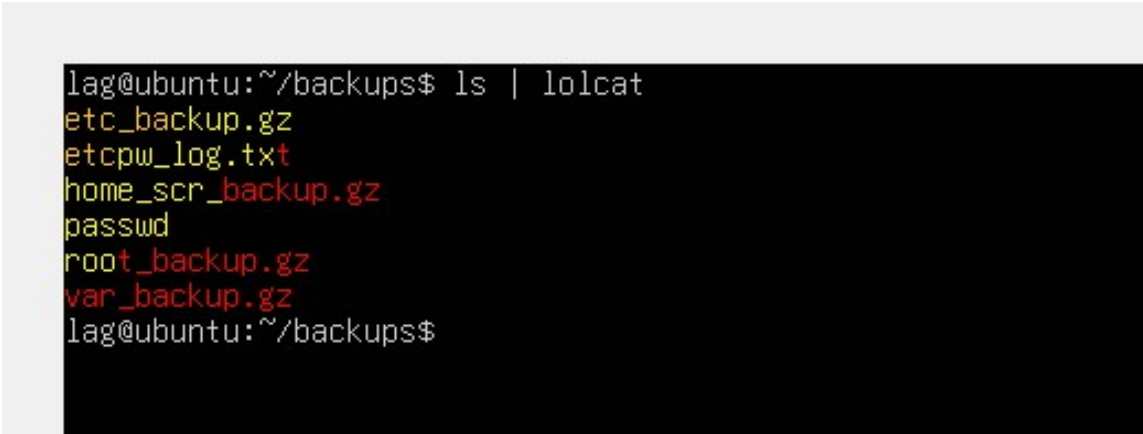
```

GNU nano 7.2                                     /home/lag/scripts/backup1.sh *
#!/bin/bash

##rsync command to create backups of /etc, /home, /root and /var directories
##save it to new backups directory

sudo rsync -avz /etc /home/lag/backups/etc_backup.gz
sudo rsync -avz /home/lag/scripts /home/lag/backups/home_scr_backup.gz
sudo rsync -avz /var /home/lag/backups/var_backup.gz
sudo rsync -avz /root /home/lag/backups/root_backup.gz

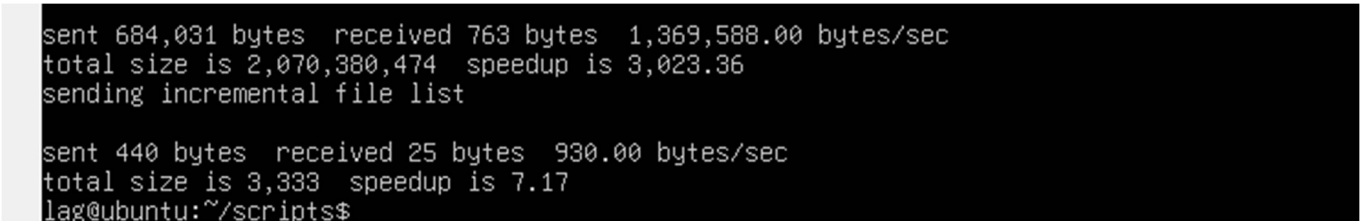
```

Example 2.2 Image of the backups in the backups directory

```
lag@ubuntu:~/backups$ ls | lolcat
etc_backup.gz
etcpw_log.txt
home_scr_backup.gz
passwd
root_backup.gz
var_backup.gz
lag@ubuntu:~/backups$
```

3. Set up a cron job to run your backup script at specific intervals (daily, weekly, and/or monthly).

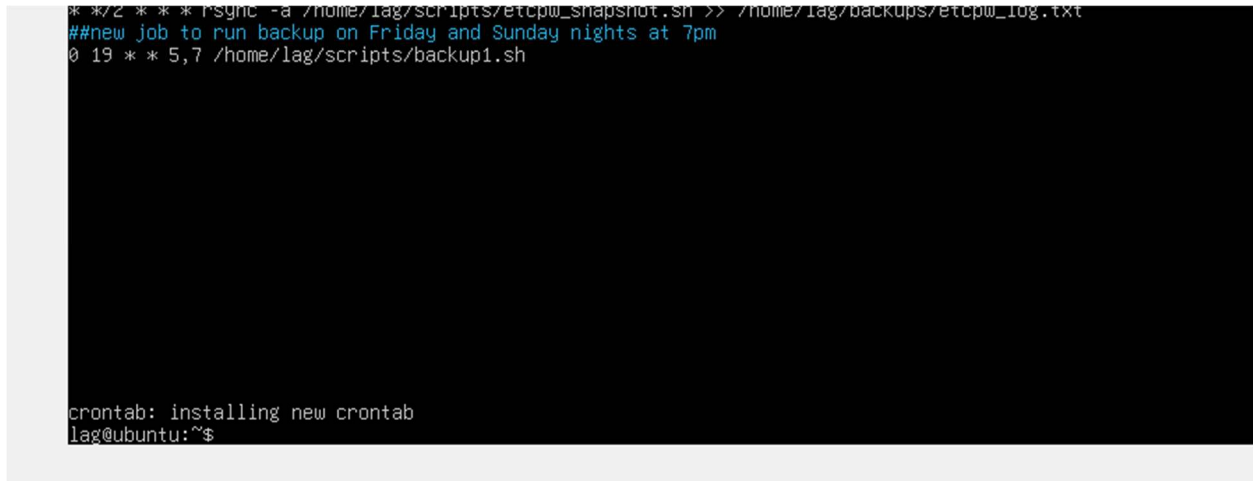
My rsync backup script updates files incrementally, so it uses the first backups as a starting point and only updates what has changed (Crape, 2024). To verify this, I ran the script again a little while after the first time it was run. On the second run, it completed much quicker since it did not have to transfer every file again, only the changes. The script output also listed information that it updated incrementally. The frequency in which a backup should run depends on how the server is used. For systems where the data changes on a regular basis, frequent backups such as daily may be appropriate. The less frequent the backups, the more likely that the backups will contain stale data. This server is used frequently on the weekends, and less so during the week. Therefore, my backup will run on Friday night and again on Sunday night. Minimal data changes happen during the week, so the Friday backup will capture and save any progress that has been made. The weekend is when most of the changes to the system happen. Because of this, the backup will run again Sunday night. Sometimes when a lot of work is being done, mistakes can be made. If there are mistakes made during the weekend, the Friday backup could be used to restore the server to the Friday state.

Figure 3.1 Image showing the output after the second time the backup script ran

```
sent 684,031 bytes received 763 bytes 1,369,588.00 bytes/sec
total size is 2,070,380,474 speedup is 3,023.36
sending incremental file list

sent 440 bytes received 25 bytes 930.00 bytes/sec
total size is 3,333 speedup is 7.17
lag@ubuntu:~/scripts$
```

Figure 3.2 Image showing the cron job that will run the backup script on Friday and Sunday at 7pm

A terminal window with a black background and white text. The text shows the process of adding a cron job. It starts with a command to add a job, followed by a comment line, and then the cron job entry itself. The prompt changes to indicate the crontab is being installed, and then returns to the user's shell prompt.

```
* */2 * * * rsync -a /home/lag/scripts/etcpw_snapshot.sh >> /home/lag/backups/etcpw_log.txt
##new job to run backup on Friday and Sunday nights at 7pm
0 19 * * 5,7 /home/lag/scripts/backup1.sh

crontab: installing new crontab
lag@ubuntu:~$
```

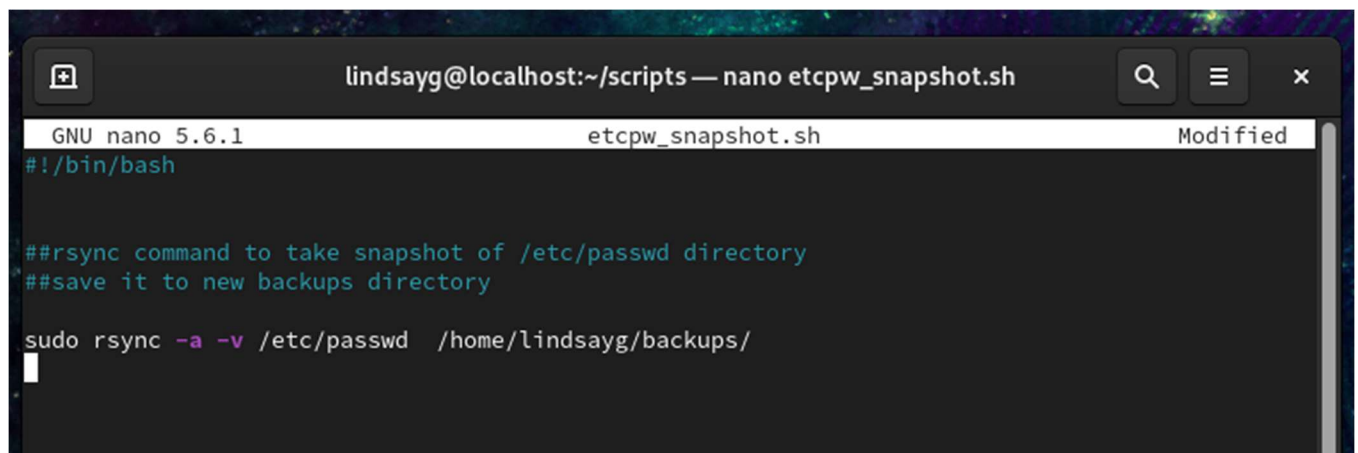
The five asterisks at the beginning of any cron job signify the minute, hour, day of month, month, and day of week (StackOverflow, 2015). To schedule the cron job, type `crontab -e` to access the configuration file. To set the time for 7pm, select a '0' for the minute, and '19' for 7pm. The next two settings would be asterisk signifying any day and any month. The final field would have '5,7' to specify both Friday (5th day of week) and Sunday (7th day of week).

CentOS Server

Example 1. Take a snapshot of users every other hour (Use a cron job for this) to see if there is any suspicious adding/removing of users

Rsync can be a useful tool for creating backups or taking snapshots of files or directories (Rubel, 2002). I chose to use rsync to monitor the `/etc/passwd` file to monitor users. First, I wrote a basic script that would copy the `/etc/passwd` file and save it to the backups directory. The `-a` option after rsync specifies archive mode, which copies files recursively and preserves permissions, timestamps, etc. (Sheldon, 2023). The `-v` option specifies that rsync should be used in verbose mode so that there is detailed information on the process.

Figure 1.1 Rsync script to take a snapshot of `/etc/passwd` directory



```
lindsayg@localhost:~/scripts — nano etcpw_snapshot.sh
GNU nano 5.6.1 etcpw_snapshot.sh Modified
#!/bin/bash

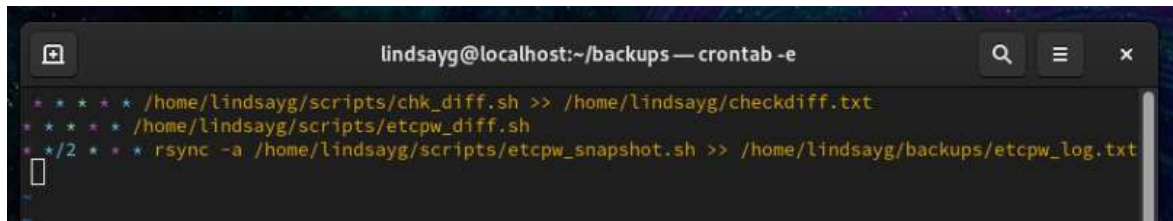
##rsync command to take snapshot of /etc/passwd directory
##save it to new backups directory

sudo rsync -a -v /etc/passwd /home/lindsayg/backups/
```

The syntax to use rsync in this script uses the command **`sudo rsync -a -v /etc/passwd(source file) /home/lindsayg/backups/(destination)`**.

Next, I set up a cron job to run the snapshot script every other hour and also to create an output file, `etcpw_log.txt`. The five asterisks at the beginning of any cron job signify the minute, hour, day of month, month, and day of week (StackOverflow, 2015). To set up the cron job, type **`crontab -e`** to go to the configuration file. The syntax to set the time portion of the job to run every other hour is **`* */2 * * *`** (Wilkinson, 2024). Then, specify the script to run, and an output file if that is desired.

Figure 1.2 Image of cron job to run rsync script



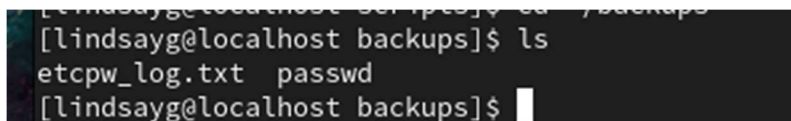
```

lindsayg@localhost:~/backups — crontab -e
* * * * * /home/lindsayg/scripts/chk_diff.sh >> /home/lindsayg/checkdiff.txt
* * * * * /home/lindsayg/scripts/etcpw_diff.sh
*/2 * * * * rsync -a /home/lindsayg/scripts/etcpw_snapshot.sh >> /home/lindsayg/backups/etcpw_log.txt

```

To confirm that everything ran as expected, navigate to the backups directory and use **ls** to see the contents. Figure 1.3 shows the snapshot of `/etc/passwd` and the log file.

Figure 1.3 Contents of backup directory after cron job



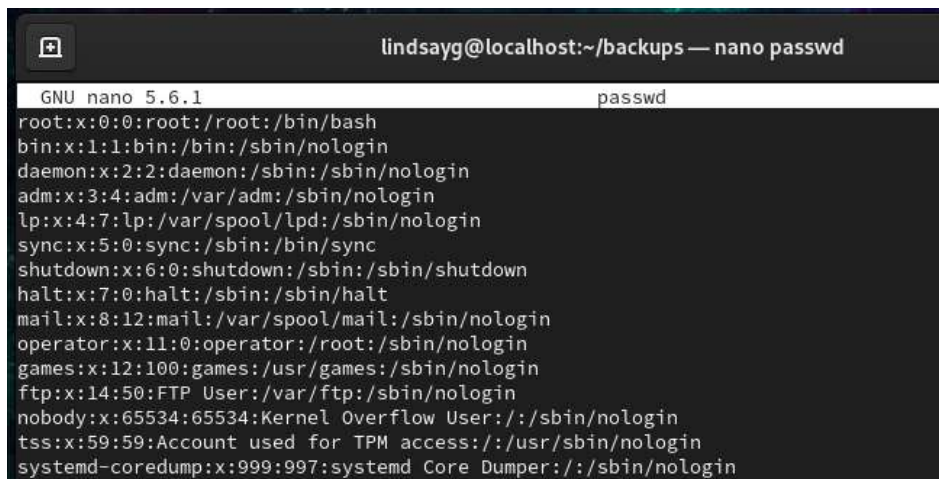
```

lindsayg@localhost:~/backups — ls
[~lindsayg@localhost backups]$ ls
etcpw_log.txt  passwd
[~lindsayg@localhost backups]$

```

To view the contents of the files, you can open them using a text editor like nano.

Figure 1.4 The contents of the passwd snapshot

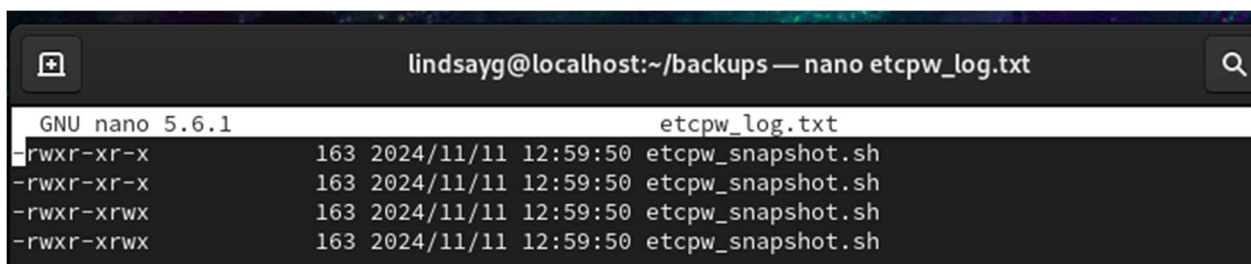


```

GNU nano 5.6.1 passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:65534:65534:Kernel Overflow User:/:/sbin/nologin
tss:x:59:59:Account used for TPM access:/:usr/sbin/nologin
systemd-coredump:x:999:997:systemd Core Dumper:/:/sbin/nologin

```

Figure 1.5 Contents of `etcpw_log.txt`



```

GNU nano 5.6.1 etcpw_log.txt
-rwxr-xr-x 163 2024/11/11 12:59:50 etcpw_snapshot.sh
-rwxr-xr-x 163 2024/11/11 12:59:50 etcpw_snapshot.sh
-rwxr-xrwx 163 2024/11/11 12:59:50 etcpw_snapshot.sh
-rwxr-xrwx 163 2024/11/11 12:59:50 etcpw_snapshot.sh

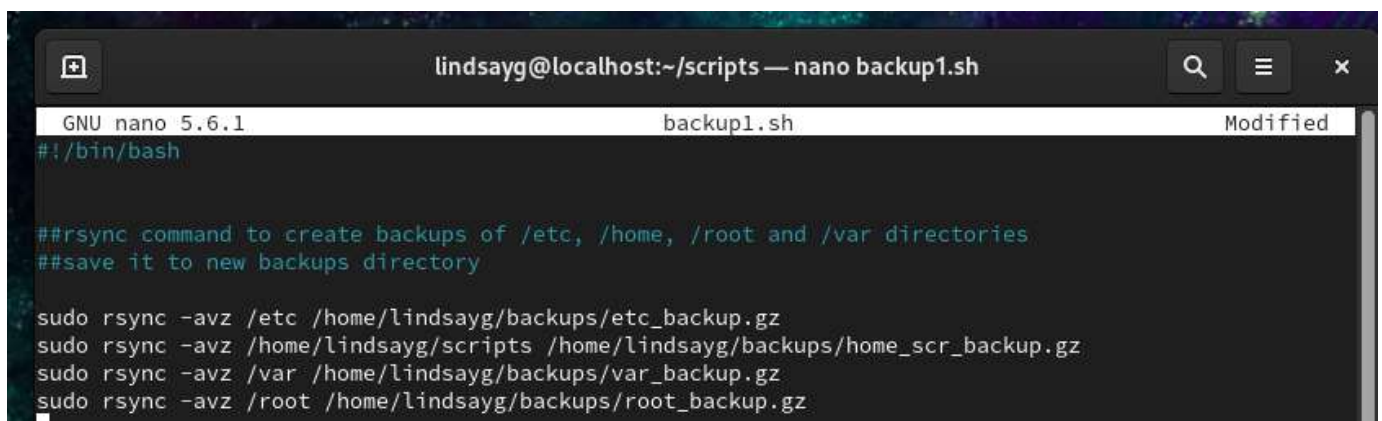
```

Note: I am unsure what the output in the log file signifies. I can't find a source that explains it, but I assume it is keeping track of when the script was run. Please note that for testing, I used all asterisks for the cron job before setting it to every other hour. That is why the time in the log does not reflect that the script has been run every other hour. I also ran this script on 11/12, yet the date shows 11/11.

Example 2. Write a script to back up your most important information and your logs

The three directories that I thought were most important to back up were the /etc, /root, /home/scripts, and /var directories. The /etc directory is important to back up because it contains all of the server's system configuration files (Hess, 2020). This includes information related to application configurations, users, passwords, and start up files. If a full restore of the server was needed, the /etc directory would be crucial for bring the system back to an operational state. The /home/scripts directory has important scripts which I as a user would not want to lose. The /var directory is important because it contains all of the logs, crontabs, and any databases that may be on a system. Lastly, the root directory would be important to backup because it holds downloads, configurations, and scripts that are unique to the system. The -z option in the backup script enables file compression when the files are being transferred (Sheldon, 2023). The snapshots are saved as compressed gzip files to save space.

Example 2.1 Script to back up /etc, /home/scripts, /root, and /var directories



```

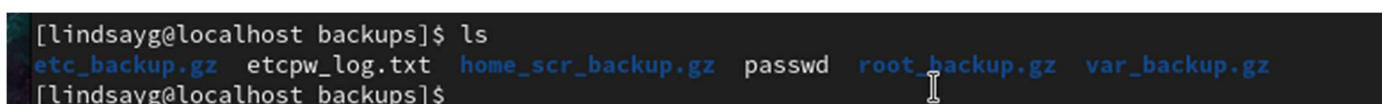
lindsayg@localhost:~/scripts — nano backup1.sh
GNU nano 5.6.1 backup1.sh Modified
#!/bin/bash

##rsync command to create backups of /etc, /home, /root and /var directories
##save it to new backups directory

sudo rsync -avz /etc /home/lindsayg/backups/etc_backup.gz
sudo rsync -avz /home/lindsayg/scripts /home/lindsayg/backups/home_scr_backup.gz
sudo rsync -avz /var /home/lindsayg/backups/var_backup.gz
sudo rsync -avz /root /home/lindsayg/backups/root_backup.gz

```

Example 2.2 Image of the backups in the backups directory



```

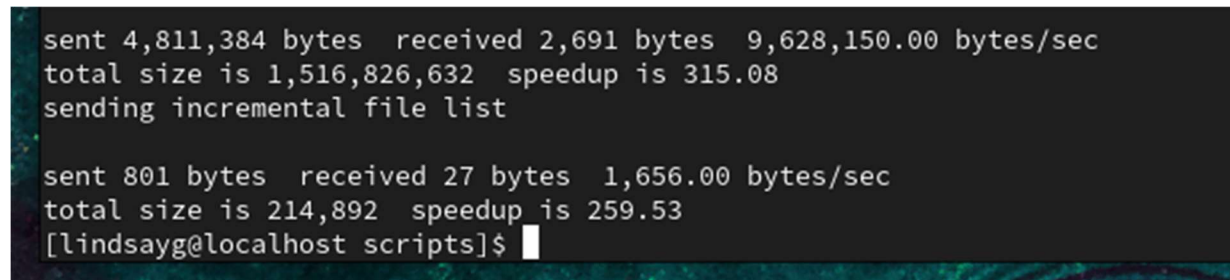
[lindsayg@localhost backups]$ ls
etc_backup.gz  etcpw_log.txt  home_scr_backup.gz  passwd  root_backup.gz  var_backup.gz
[lindsayg@localhost backups]$

```


3. Set up a cron job to run your backup script at specific intervals (daily, weekly, and/or monthly).

My rsync backup script updates files incrementally, so it uses the first backups as a starting point and only updates what has changed (Crape, 2024). To verify this, I ran the script again a little while after the first time it was run. On the second run, it completed much quicker since it did not have to transfer every file again, only the changes. The script output also listed information that it updated incrementally. The frequency in which a backup should run depends on how the server is used. For systems where the data changes on a regular basis, frequent backups such as daily may be appropriate. The less frequent the backups, the more likely that the backups will contain stale data. This server is used frequently on the weekends, and less so during the week. Therefore, my backup will run on Friday night and again on Sunday night. Minimal data changes happen during the week, so the Friday backup will capture and save any progress that has been made. The weekend is when most of the changes to the system happen. Because of this, the backup will run again Sunday night. Sometimes when a lot of work is being done, mistakes can be made. If there are mistakes made during the weekend, the Friday backup could be used to restore the server to the Friday state.

Figure 3.1 Image showing the output after the second time the backup script ran



```
sent 4,811,384 bytes  received 2,691 bytes  9,628,150.00 bytes/sec
total size is 1,516,826,632  speedup is 315.08
sending incremental file list

sent 801 bytes  received 27 bytes  1,656.00 bytes/sec
total size is 214,892  speedup is 259.53
[lindsayg@localhost scripts]$
```

Figure 3.2 Image showing the cron job that will run the backup script on Friday and Sunday at 7pm



```
@ 19 * * 5,7 /home/lindsayg/scripts/backup1.sh
```

The five asterisks at the beginning of any cron job signify the minute, hour, day of month, month, and day of week (StackOverflow, 2015). To schedule the cron job, type `crontab -e` to access the configuration file. To set the time for 7pm, select a '0' for the minute, and '19' for 7pm. The next two

settings would be asterisk signifying any day and any moth. The final field would have '5,7' to specify both Friday (5th day of week) and Sunday (7th day of week).

References

Crape, M. (2024, January 11). *Guide to Server Backups: Creating a Backup Strategy* | Veeam. Veeam Software Official Blog. <https://www.veeam.com/blog/server-backup-guide.html>

Hess, K. (2020, December 15). *9 Linux directories you must back up and one you shouldn't*. Redhat.com. <https://www.redhat.com/en/blog/backup-dirs>

Kiarie, J. (2020, December 8). *How to Boot into Rescue Mode Or Emergency Mode In Ubuntu 20.04 / 18.04*. Wwww.tecmint.com. <https://www.tecmint.com/boot-ubuntu-in-rescue-mode/>

Kiarie, J. (2024, June 13). *Create and Run Your First Shell Script in Linux DevOps Essentials Tutorial*. LinuxTechi. <https://www.linuxtechi.com/boot-ubuntu-24-04-into-rescue-mode/>

Los-Merengue. (2023, January 20). *Linux Daemon Configuration - Los-Merengue - Medium*. Medium; Medium. <https://medium.com/@Los-merengue/linux-daemon-configuration-c07e4eda3f37>

Matade, S. (2023, June 21). *Creating and Managing Daemon Services in Linux using Systemd*. Medium; Medium. <https://medium.com/@shreyasmatade/creating-and-managing-daemon-services-in-linux-using-systemd-9ea9ac1fbf37>

Mathew, N. (2021, March 26). *Welcome to Emergency mode in Linux - Boot error*. Bobcares. <https://bobcares.com/blog/welcome-to-emergency-mode-in-linux-boot-error/>

Medium. (2020, April 5). *Get grub menu back after installing Ubuntu 20.04 alongside Windows*. Medium. <https://medium.com/@leijerry888/get-grub-menu-back-after-installing-ubuntu-20-04-alongside-windows-dab5de5afc37>

Red Hat. (2023). *Chapter 12. Managing systemd* | Red Hat Product Documentation. Redhat.com. https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/9/html/configuring_basic_system_settings/managing-systemd_configuring-basic-system-settings#listing-system-services_managing-system-services-with-systemctl

Red Hat. (2024). *33.3. Emergency Mode* | Red Hat Product Documentation. Redhat.com. https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/6/html/deployment_guide/sec-emergency_mode

Rubel, M. (2002). *www.mikerubel.org*. Cornell.edu.

http://bigbro.biophys.cornell.edu/internal_info/computer/FileManagement/rsync_backups.html

Server World. (2022). *CentOS Stream 9 : Initial Settings : Services : Server World*. Server-World.info.

https://www.server-world.info/en/note?os=CentOS_Stream_9&p=initial_conf&f=4

Sheldon, K. (2023, April 12). *Using rsync for Backups on Linux/Unix Systems*. Nexcess.

<https://www.nexcess.net/help/using-rsync-for-backups/>

StackOverflow. (2015, July 7). *How to run a cron job on every Monday, Wednesday and Friday?* Stack Overflow. <https://stackoverflow.com/questions/31260837/how-to-run-a-cron-job-on-every-monday-wednesday-and-friday>

Wilkinson, P. (2024, January 31). *How to run cron every hour - with variations and examples | Warp*.

Warp. <https://www.warp.dev/terminus/how-to-run-cron-every-hour>