Contingency Planning and Backups: SSH/SCP, Deltas, and Rsync

License

Contents

Preparation

These tasks can be completed on the LinuxZ IMS image.

This lab could be completed on most Linux systems with rsync and sshd installed.

Introduction

As you will recall, availability is a common security goal. This includes data availability, and systems and services availability. Preparing for when things go wrong, and having procedures in place to respond and recover is a task known as contingency planning. This includes business continuity planning (BCP), which has a wide scope covering many kinds of problems, disaster recovery planning, which aims to recover ICT after a major disaster, and incident response (IR) planning, which aims to detect and respond to security incidents.

Business impact analysis involves determining which business processes are mission critical, and what the recovery requirements are. This includes Recovery Point Objectives (RPO), that is, which data and services are acceptable to lose and how often backups are necessary, and Recovery Time Objectives (RTO), which is how long it should take to recover, and the amount of downtime that is allowed for.

Having reliable backups and redundancy that can be used to recover data and/or services is a basic security maintenance requirement.

Uptime

At a console, run:

```
uptime
```

A common goal is to aim for “five nines” availability (99.999%). If you only have one server, that means keeping it running constantly, other than for scheduled maintenance. List a few legitimate security reasons for performing off-line maintenance:

Copy

The simplest of Unix copy commands, is “cp”. Cp takes a local source and destination, and can recursively copy contents from one file or directory to another. Run:

```
cp /etc/passwd /tmp
```

We have made a backup of a source file (/etc/passwd), to our destination directory (/tmp). Note that we lost the metadata associated with the file, including file ownership and permissions:

```
ls -la /tmp/passwd
```

SSH/SCP

Using SSH (secure shell), scp (secure copy) can transfer files securely (encrypted) over a network. This replaces the insecure rcp command, which sends files over the network in-the-clear (not encrypted). Rcp should never be used.

Working with a classmate, start by logging in to the classmates system via ssh:

If you wish, you can use Wireshark to confirm that the scp traffic is encrypted.

Backup your /etc/ directory to their computer using scp:

> `sudo scp -pr /etc/ username@their-ip-address:/home/username/ssh_backup/`
>
> If you are working alone, you can simply start two computers or VMs (with sshd running), and run the above.
>
> This may take some time, feel free to open another terminal console, to read the scp man page

Read the scp man page to determine what the -p and -r flags do.

> Hint: “man scp”, press “q” to quit.

Now, lets add a file to /etc, and repeat the backup:

> `sudo bash -c 'echo hello > /etc/hi'`
>
> `sudo scp -pr /etc/ username@their-ip-address:/home/username/mybackup/`

Note that the program re-copies all of the files entirely, regardless of whether (or how much) they have changed.

Ssh to their computer, to look at your backup files:

> `ssh username@their-ip-address`

This will log you in with *username* on their system. Assuming their computer has the same user account available (as is the case with the LinuxZ image), you can omit “username”, and just run “ssh their-ip-address”, and you will be prompted to provide authentication for your own account, as configured on their system.

> `tree /home/username/ssh_backup/`

Exit ssh:

```
exit
```

(Or Ctrl-D)

Note, this command will close your bash shell, if you are not logged in via ssh; so be careful.

Save a screenshot of the scp backup, as evidence that you have completed this part of the task.

Label it or save it as "Backup-1".

Rsync, deltas and epoch backups

Rsync is a popular tool for copying files locally, or over a network. Rsync can use delta encoding (only sending *differences* over the network rather than whole files) to reduce the amount of data that needs to be transferred. Many commercial backup systems provide a managed frontend for rsync.

Note: make sure you exited ssh above, and are now running commands on your local system.

Lets start by doing a simple copy of your /etc/ directory to a local copy:

```
sudo rsync -av /etc /tmp/etc_backup/
```

Note that the presence of a trailing "/" changes the behaviour, so be careful when constructing rsync commands. In this case we are copying the directory (not just its contents), into the directory etc_backup. See the man page for more information.

Rsync reports the amount of data "sent".

Read the rsync man page, to understand the flags we have used (-a and -v). As you will see, Rsync has a great deal of options.

Now, lets add a file to /etc, and repeat the backup:

```
sudo bash -c 'echo hello > /etc/hello'

sudo rsync -av /etc /tmp/etc_backup/
```

Note that only the new file was transferred (incremental) to update our epoch (full) backup of /etc/.

Rsync remote copies via SSH with compression

Rsync can act as a server, listening on a TCP port. It can also be used via SSH, as you will see. Copy your /etc/ directory to your classmate's computer using Rsync via SSH:

```
sudo rsync -avzh /etc username@their-ip-address:/home/
username/my-rsync-backup/
```

Tip: this is all one line, replace *username*, and *their-ip-address*

Note that this initial copy will have used less network traffic compared to scp, due to the -z flag, which tells rsync to use compression. Compare the amount of data sent (as reported by rsync in the previous command -- the -h told rsync to use human readable sizes) to the size of the date that was sent:

```
sudo du -sh /etc
```

Now, if you were to delete a local file that had been backed up:

```
sudo rm /etc/hello
```

Even if you resync your local changes to the server (your classmate's system), the file will not be deleted from the server:

```
sudo rsync -avz /etc username@their-ip-address:/home/
username/my-rsync-backup/
```

Tip: this is all one line

To recover the file, you can simply retrieve the backup:

```
sudo rsync -avz username@their-ip-address:/home/username/
my-rsync-backup/etc/hello /etc/
```

Tip: this is all one line

Note, however, that the new file is now no longer owned by root. This can be avoided, if you have SSH root access to the remote machine (for security reasons this is not usually done) to retain ownership and so on. Alternatively, you can use an Rsync server running as root.

Delete the file locally, and sync the changes *including deletions* to the server so that it is also deleted there:

```
sudo rm /etc/hello
```

```
sudo rsync -avz --delete /etc username@their-ip-address:/home/username/my-rsync-backup/
```

Confirm that the file has been deleted from the backup stored on the server (your classmate's system).

Compare the file access/modification times of the scp and rsync backups, are they the same/similar? If not, why? Note that rsync can retain this information, if run as root.

Rsync incremental/differential backups

If you need to keep daily backups, it would be an inefficient use of disk space (and network traffic and/or disk usage) to simply save separate full copies of your entire backup each day. Therefore, it often makes sense to copy only the files that have changed for our daily backup. This can either be comparisons to the last backup (incremental), or last full backup (differential).

Differential backups

Create a full backup of /etc/ to your local /tmp/ directory.

Create a new file in /etc:

```
sudo bash -c 'echo "hello there"> /etc/hello'
```

And now lets create a differential backup of our changes to /etc:

```
sudo rsync -av /etc --compare-dest=/tmp/etc_backup/ /tmp/etc_backup2/
```

The --compare-dest flag tells rsync to search these backup copies, and only copy files if they have changed since a backup. Refer the the man page for further explanation.

Look at what is contained in /tmp/etc_backup2/:

```
tree /tmp/etc_backup2

ls -la /tmp/etc_backup2/
```

Note that there are lots of empty directories, with only the files that have actually changed (in this case /etc/hello).

Now create another change to /etc:

```
sudo bash -c 'echo "hello there!"> /etc/hi'
```

To make a differential backup (saving changes since the last full backup), we just repeat the previous command, with a new destination directory:

```
sudo rsync -av /etc --compare-dest=/tmp/etc_backup/ /tmp/
etc_backup3/
```

Look at the contents of your new backup. You will find it now contains your two new files. That is, all of the changes since the full backup.

Delete a non-essential existing file in /etc/, and our test hello file:

```
sudo rm /etc/wgetrc /etc/hello
```

Now restore from your backups by first restoring from the full backup, then the latest differential backup. The advantage to a differential backup, is you only need to use two commands to restore your system.

Save a screenshot(s) of the commands you used to restore your differential backups (run "history|tail"), as evidence that you have completed this part of the task.

Label it or save it as "Backup-2".

Incremental backups

The disadvantage of the above differential approach to backups, is that your daily backup gets bigger and bigger every day until you do another full backup. With incremental backups you only store the changes since the last backup.

Now create another change to /etc:

```
sudo bash -c 'echo "Another test change"> /etc/test1'
```

Now create an incremental backup based on the last differential backup:

```
sudo rsync -av /etc --compare-dest=/tmp/etc_backup/
--compare-dest=/tmp/etc_backup3/ /tmp/
etc_backup_incremental1/
```

Another change to /etc:

```
sudo bash -c 'echo "Another test change"> /etc/test2'
```

Now create an incremental backup based on the last differential backup and the last incremental backup:

```
sudo rsync -av /etc --compare-dest=/tmp/etc_backup/ -
-compare-dest=/tmp/etc_backup3/ --compare-dest=/tmp/
etc_backup_incremental1/ /tmp/etc_backup_incremental2/
```

If we were to delete a number of files:

```
sudo rm /etc/wgetrc /etc/hello /etc/test1 /etc/test2
```

Now restore /etc by restoring from the full backup, then the last differential backup, then the first incremental backup, then the second incremental backup.

Save a screenshot(s) of the commands you used to restore your incremental backups (run "history|tail"), as evidence that you have completed this part of the task.

Label it or save it as "Backup-3".

Rsync snapshot backups

Another approach to keeping backups is to keep a snapshot of all of the files, but wherever the files have not changed, a hard link is used to point at a previously backed

up copy. If you are unfamiliar with hard links, read more about them online. This approach gives users a snapshot of how the system was on a particular date, without having to have redundant full copies of files.

These snapshots can be achieved using the --link-dest flag. Open the Rsync man page, and read about --link-dest. Lets see it in action.

Make another copy of our local /etc backup:

```
sudo rsync -avh --delete /etc /tmp/etc_snapshot_full
```

And make a snapshot containing hard links to files that have not changed, with copies for files that have changed:

```
sudo rsync -avh --delete --link-dest=/tmp/etc_snapshot_full
/etc /tmp/etc_snapshot2
```

Rsync reports not having copied any new files, yet look at what is contained in /tmp/etc_snapshot2. It looks like a complete copy, yet is taking up almost no extra storage space.

Create other changes to /etc:

```
sudo bash -c 'echo "Another test change"> /etc/test3'
```

```
sudo bash -c 'echo "Another test change"> /etc/test4'
```

And make a new snapshot, with copies of files that have changed:

```
sudo rsync -avh --delete --link-dest=/tmp/etc_snapshot_full
/etc /tmp/etc_snapshot2
```

Delete some files, and and make a new differential snapshot. Although Rsync does not report a deletion, the deleted files will be absent from the new snapshot. Recover a file from a previous snapshot.

Save a screenshot of the commands you used to delete some files, take a differential rsync snapshot, and recover the files from a previous snapshot backup (run “history|tail”), as evidence that you have completed this part of the task.

Label it or save it as “Backup-4”.

Problem-based tasks

Configure a system for automatic daily Rsync **differential** backups, using crontab, and possibly a backup script of your own design

Helpful resource:

> http://webgnuru.com/linux/rsync_incremental.php

Save a screenshot of the commands/configuration you used to create automatic daily Rsync differential backups, using crontab, and possibly a backup script of your own design, as evidence that you have completed this part of the task.

Label it or save it as "Backup-A1".

Configure a system for automatic daily Rsync **incremental** backups, using crontab, and possibly a backup script of your own design

---

Save a screenshot of the commands/configuration you used to create automatic daily Rsync incremental backups, using crontab, and possibly a backup script of your own design, as evidence that you have completed this part of the task.

Label it or save it as "Backup-A2".

Configure a system for automatic daily Rsync **differential snapshot** backups, using crontab, and possibly a backup script of your own design

---

Save a screenshot of the commands/configuration you used to create automatic daily Rsync differential snapshot, using crontab, and possibly a backup script of your own design, as evidence that you have completed this part of the task.

Label it or save it as "Backup-A3".

Configure a system for automatic daily Rsync **incremental snapshot** backups, using crontab, and possibly a backup script of your own design

---

Save a screenshot of the commands/configuration you used to create automatic daily Rsync incremental snapshot, using crontab, and possibly a backup script of your own design, as evidence that you have completed this part of the task.

Label it or save it as "Backup-A4".

Set up an Rsync server (rsyncd), and configure a module for read-write access, then write files to the server.

Hints:

- change /etc/rsyncd.conf
- start rsyncd
- ensure the firewall allows access to the rsync port
- write to the module using rsync from a separate computer

Helpful resource:

http://everythinglinux.org/rsync/

Save a screenshot of the commands/configuration you used to start an rsync server with a module for read-write access and then write files to the server, as evidence that you have completed this part of the task.

Label it or save it as “Backup-A5”.

What to show your tutor for XP rewards

Show your tutor each of the above (in red) evidences. You may be asked to justify your decisions. This will be used to allocate XP for the module. Further details of the XP rewards and requirements are available on the *My XP* site.