Make off-site backups or you will lose your data

BACKUP PAINS

Who needs attackers when you have system administrators? Learn why copying your data doesn't mean you've backed it up. BY KURT SEIFRIED

As I write this column, I cannot help but reflect on the irony of just having wiped out a month's worth of data. In the spirit of this article, I was fiddling around with backups on my web server, and I managed to accidentally delete most of */var/* and all of the */home/* directory. This wouldn't have been so bad if I hadn't kept the daily backups in */home/backups/*. Oops.

Backing Up Doesn't Always Mean You Have Backups

If your data isn't available, or the systems to process and serve it aren't available, you have a problem. In the case of my web server, the missing */var/* and */home/* render it pretty much useless. It serves 404s and that's pretty much it. To make sure data is available, you need to back it up. Seems simple right? In reality most of us (myself included) get it wrong, and although we go through the motions of making a backup, what we're really doing is just copying the data somewhere else that is equally vulnerable to loss.

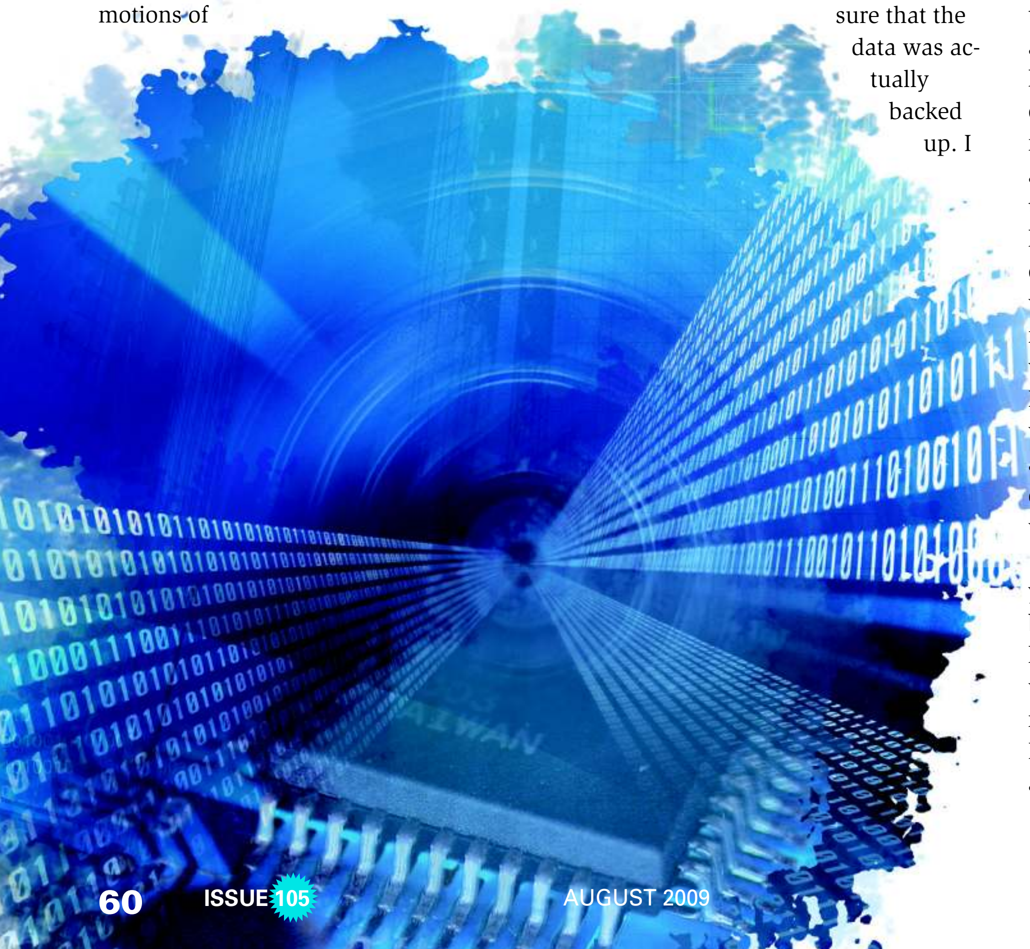
In my case, I made a classic mistake of storing my backups on the same system that the data being backed up is on, and to make things worse, I actually kept it in a commonly accessed directory. Not that this would have mattered. Because the server only has one hard drive, I am only a single disk failure away from complete data loss no matter how much I back my data up locally on the server. Even if I were to install a second hard drive in the machine, it's still all too easy for a single event (bad drive controller, attacker wiping the system, fire, flood, power supply going bonkers, theft, etc.) to wipe out more than one hard drive.

What Are Real Backups?

Three main elements go into making real backups. Number one: You have to ensure that the data was actually backed up. I have seen far too many systems that write the data to a CD, DVD, or tape improperly, which results in nonrecoverable data. Ideally, you need to test out every backup you make, but if this isn't practical, you at least need to make occasional spot checks to ensure the data can be recovered.

Number two: You need to have off-site backups that are as close to read-only as you can get. This doesn't necessarily mean they have to be in a different physical location (although this is always a good idea), but they have to at least be separate enough that a single failure or event such as formatting a disk array or losing a server won't wipe out both the live data and the backups. A perfect example of this in action (besides, of course, my recent faux pas) is the website AVSIM Online, which lost 13 years of data to a single attack [1]. According to reports, AVSIM Online had two servers that copied their data off each other in an effort to back each other up. As I have said before, many of us are only copying data when we do backups and not making actual backups. In this case, an attacker broke into both servers, since they were basically identical, and deleted all the live data and the copies held on both servers. AVSIM Online lost their website, email, file library, forums, and more in one fell swoop and most likely will never get the data back. In my case, I was lucky. I only deleted a month's worth of logfiles and collected data, so all I have to do is wait a month for new data to be collected – good thing this wasn't someone's financial records.

Number three: You need to make sure you aren't deleting files out of the backup or zeroing the file contents unless you are beyond 100% certain you will never need that file again. For this reason, RAID isn't a backup solution. Even if you have multiple hard drives in a RAID configuration so that the loss of



a single or even multiple drives will not-cause data loss, you can still lose data through deletion (*rm*, *mkfs*, etc.) or zeroing or altering of files (*cat foo > bar*).

Get Off the System

Fortunately almost every mature backup program supports getting data from a client and storing it somewhere else – often on a dedicated server, disk array, tape, DVD, and so on. Some excellent options exist for Linux: Amanda [2], which ships with almost every distribution, as well as BackupPC [3] and Bacula [4], both covered in *Linux Pro Magazine* [5] [6]. Although I won't cover the details here, suffice it to say they are very powerful, have lots of knobs to adjust, and definitely back up your data if you set them up right. The quick and dirty option is rsync (*yum install rsync*, *apt-get install rsync*, etc.) [7].

The Problem with rsync

Rsync was designed to keep large sets of files synchronized between different systems or directories. As such, it does a pretty good job as a poor man's backup tool or helper tool. Typically I do local backups on the system with *tar* and *mysqldump*, putting the files into a directory with a date stamp (you'll see why later).To use rsync, simply create the file *rsyncd.conf* in */etc/*:

```
uid = backups
gid = backups
use chroot = yes
[backups]
        path = /backups/
        read only = yes
```

Now enable it in inetd or xinetd. On the client side, you use a command such as *rsync -a 10.1.2.3::backups/* /mybackups/* to copy the contents from the directory */backups/* on the remote server 10.1.2.3 to your local */mybackups/* directory. Now to make really sure nothing bad happens, you'll notice I didn't use the *--delete* option, which allows rsync to delete local files that are no longer present on the remote end. What could possibly go wrong? If a file gets deleted on my server, I'll still have a local copy. Right? Yes, but if a file gets zeroed by an attacker or from an error in a backup script (e.g., *cat 0 > somefile*), then the local copy will also get zeroed. In other words, say goodbye to your data. The solution is to make incremental backups, which is why I place my daily backups in a directory with that day's date as the name and then rsync that directory only:

```
rsync -a ⮐
10.1.2.3::backups/`date +%Y-%m-%d` ⮐
/my-backups/
```

Anything in back ticks is executed by the shell and the output used. This means that in a worst case scenario I might lose today's backups, but I should never lose older backups because they are in a separate directory on my local server that I don't write to with rsync.

Do You Know Where Your Backups Are?

So you now have daily backups, copied off the server onto another hopefully secure machine. Or do you? All too often automated programs fail, IP addresses and hostnames change, the rsync configuration gets modified, the local disk partition for backups gets filled up, or who knows what happens. The final piece to the backup puzzle is automated notification of whether or not the backup was successful. The programs mentioned above (Amanda, Bacula, etc.) all support notification, but how can you build notification into your home-built rsync system? The solution to this is simple and elegant: Just add the following line to your backup script:

```
ls -la /my-backups/`date +%Y-%m-%d` ⮐
| mail -s "daily backup report" ⮐
your@email.address
```

This will do a directory listing of the new backup directory and pipe the output to the mail command, sending you an email with a listing of all the files and their sizes. In fact, you can go farther and list files within the archives with the *tar -t* command if you want to get fancy.

In most cases, if you run rsync with the *-v* (verbose) option, you will receive output such as:

```
receiving file list ... done
2009-05-27/
2009-05-27/home.tar.bz2
2009-05-27/etc.tar.bz2
```

Please note that if you run commands from crontab, the output will automatically be emailed to the user that runs it.

Testing Your Backups

Some would say this final step is the most important when doing backups. First you need to take your backups and unpack them, load the tape onto a system, or do whatever you would actually do with them if you needed to restore data; otherwise, how can you be sure they work? Once you have done this, you can sleep soundly knowing that natural disasters (or ham-fingered system administrators) won't ruin your day. ■

Principles of Security

The three principles of security are: Availability, Integrity, and Confidentiality (also referred to as the AIC triad). In a nutshell, you have to keep the stuff you need to work working, you need to ensure that your data hasn't been changed by an attacker, and you need to keep your private stuff confidential.

INFO

[1] "Avsim.com Hacked – Total Data Loss" by James Anderson, *Tech Fragments*, May 15, 2009, *http://techfragments.com/news/769/Tech/Avsim-com_Hacked_-_Total_Data_Loss.html*

[2] Amanda: *http://www.amanda.org/*

[3] BackupPC: *http://backuppc.sourceforge.net/*

[4] Bacula: *http://www.bacula.org/*

[5] "BackupPC" by David Nalley, *Linux Pro Magazine*, August 2008, *http://www.linux-magazine.com/w3/issue/93/072-075_backup.pdf*

[6] "Bacula" by Jens-Christoph Brendel and Marc Schöchlin, *Linux Pro Magazine*, August 2005, *http://www.linux-magazine.com/w3/issue/57/Bacula_Backup_System.pdf*

[7] rsync: *http://samba.anu.edu.au/rsync/*

THE AUTHOR

Kurt Seifried is an Information Security Consultant specializing in Linux and networks since 1996. He often wonders how it is that technology works on a large scale but often fails on a small scale.