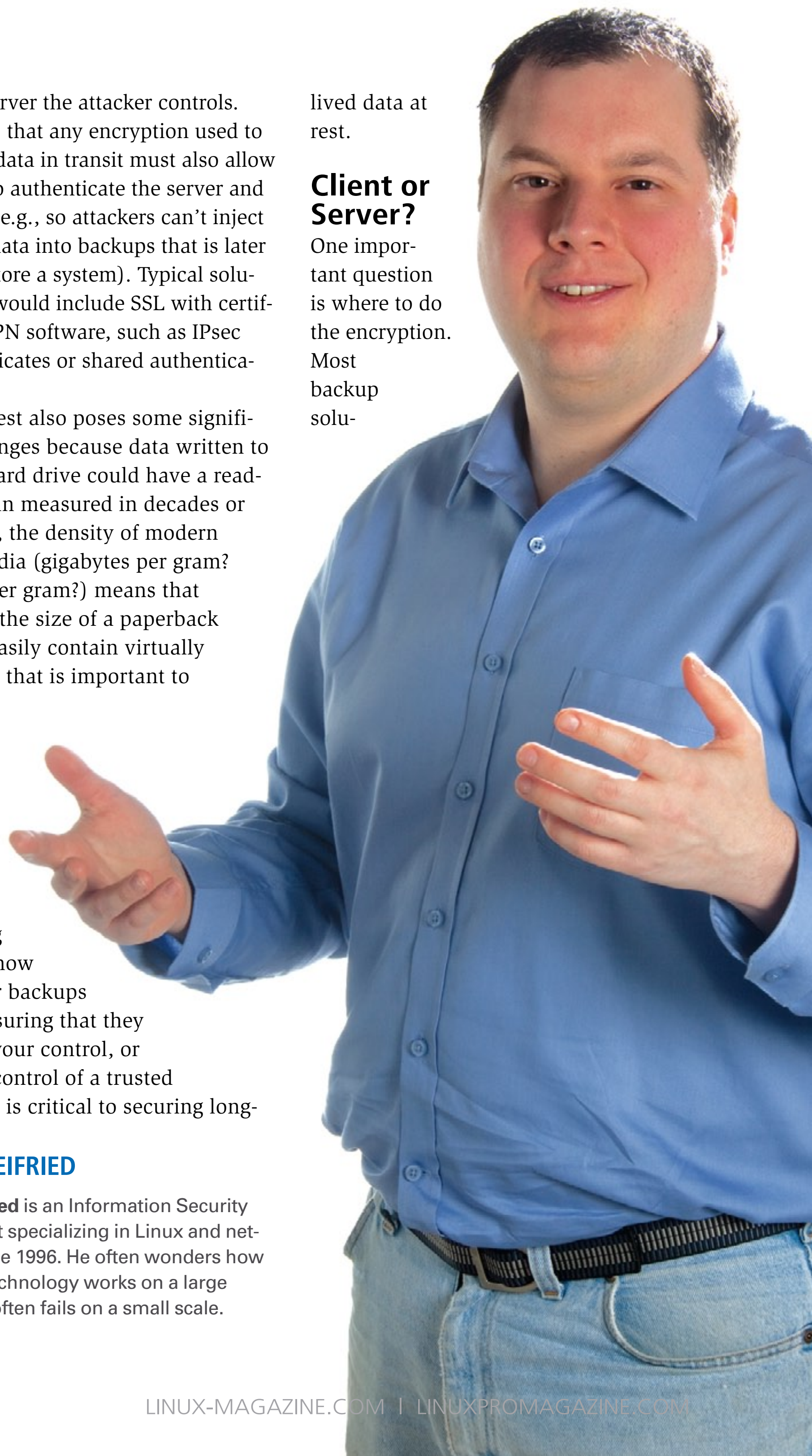
Back up your systems securely

System Rescue

Creating backups is one thing, making sure they're secure is another. We offer some tips for ensuring the process is as painless as possible. By Kurt Seifried



Making backups of your data is critical. If you don't regularly create usable backups, any outages, disk failures, or administrative errors can cause permanent data loss. But, although backups address the availability (and to some degree integrity) aspects of the AIC security triad (availability, integrity, confidentiality), they can introduce significant risks with respect to the confidentiality or secrecy of your data. In other words, when you centralize all your data on removable storage (e.g., tape drives), things can get very bad very quickly if the tapes are misplaced or stolen.

Encrypted Backups

The solution, of course, is to encrypt your backups. Depending on the risk you're willing to accept, strong encryption can even let you store your files in potentially insecure locations (e.g., a public cloud storage provider). In general, most backup programs support 256-bit AES, which is extremely strong, so with properly generated keys that are secured from attackers, your data should be safe from decryption for at least a few decades. However, you must consider several other issues when deciding how to apply encryption to your backups and how to manage your encryption keys.

Data in Transit and at Rest

Generally speaking, data is considered either "in transit" (being sent over a network) or "at rest" (having been written to media such as a tape or a hard drive). Data in transit is vulnerable to interception – even local area networks can be compromised and configured to mirror data to a server the attacker controls. This means that any encryption used to secure the data in transit must also allow the client to authenticate the server and vice versa (e.g., so attackers can't inject malicious data into backups that is later used to restore a system). Typical solutions here would include SSL with certificates or VPN software, such as IPsec using certificates or shared authentication.

Data at rest also poses some significant challenges because data written to a tape or hard drive could have a readable lifespan measured in decades or more. Also, the density of modern storage media (gigabytes per gram? terabytes per gram?) means that something the size of a paperback book can easily contain virtually all the data that is important to you.

Inventory control is a feature that is often ignored, but making sure you know where your backups are and ensuring that they are under your control, or under the control of a trusted third party, is critical to securing long-lived data at rest.

Client or Server?

One important question is where to do the encryption. Most backup solu-

KURT SEIFRIED

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.

tions are now client/server based – because almost everyone has more than one server to back up, so you need some centralized management and control to keep everything working properly. If you encrypt on the client, the data is encrypted immediately and the risk of exposure is minimized. The downside, however, is that verifying data on the server will require the private keys to be present, which can create a potential target for attackers. Encrypting the data on the server centralizes key management and allows you to perform deduplication more easily (things like `/bin/` will be the same across servers with the same OS and version installed).

Key Management

If you lose the keys to your encrypted data or the passphrases used to unlock them, you will also lose access to your data. This also means that if the people that know the passphrases die or leave the company, you might also lose access to your data. In a nutshell, you want more than one person to have access to the data. Splitting up the passphrase (e.g., so one person types in their half and a second person types in the other half) can also prevent a single person from selling all your data to a competitor. Rotating keys is also a good idea, partly because it helps enforce policies to deal with key changes, which will happen as people join and leave your organization, but also because the more data encrypted using a single key the more severe any compromise of that key will be. Finally, certain legal requirements can affect retention and access to the encryption keys, so consult a lawyer if needed. (These requirements mostly apply to financial, health, and personal information, but they differ widely by country, so you need to be informed.)

Types of data

Before I talk about backup software, I want to mention briefly the different types of data that need to be backed up. The first (and simplest) type is simply where you grab the entire file. This approach works well for files that are not modified a lot (e.g., system binaries and libraries).

The next type is structured data, such as databases, NoSQL services, mail spool files, and so on. The challenge here is that you typically cannot directly grab the files because they are unlikely to be in a consistent state (data might be waiting to be written, transactions could be half finished, etc.). In these cases, shutting down the service and then grabbing the file is also rarely an option because outages are not possible. This means your backup software will actually need to talk to the application. For the more popular applications (like most databases), finding backup software that can talk to them and extract the data shouldn't be too hard. However, for a lot of modern NoSQL and "big data" solutions, backups can still be a challenge, so you might want to think about architecting your software to make backups possible (e.g., writing object data to a logfile or sending objects to a backup server to be archived).

Backup Software

In the Linux world, you have several main options for backup software. Two of the most popular are Amanda [1] and Bacula [2]. Both are open source, and Amanda is backed by Zmanda, which builds a commercial version of Amanda with additional capabilities. Bacula has support for SQLite, MySQL, and PostgreSQL. Amanda also can support these, but you'll basically need to create some scripts to dump the databases first.

Of course, hundreds of other open source backup programs are available – most of which aren't very good (i.e., they lack polish, features, etc.) – so, in general, I advise sticking to Amanda or Bacula.

Bacula also offers limited file deduplication now; however, the functionality is relatively simplistic: You specify a "base" backup (e.g., of your standard server) and then back up additional servers that are running the same software, and Bacula will deduplicate at the file level. The good news is that several filesystems now offer deduplication support, such as Opendedup [3], so this technology should be rolled into Amanda and Bacula at some point.

NoSQL Backup

Although almost all NoSQL services have built-in redundancy and distribution of data, this feature won't help if you make an administrative error and delete a whole bunch of data or if your entire data center gets flooded (as happened during Hurricane Sandy recently). With NoSQL systems, you typically have three options: back up the filesystem (e.g., using LVM snapshots), replicate the data off site, and use a specific dump tool to extract all the data.

If you plan to use filesystem backups to grab the data, you'll need to make sure any data stored in memory is also copied onto the filesystem. If you plan to replicate the data off site, you'll need enough bandwidth with low enough latency to ensure the data gets there. Finally, for specific dumping tools, you'll generally need either to stop the service or put it into read-only mode so you can make a copy of the data that is consistent and correct. This last option, of course, has a significant effect on system availability if you need to disable writes for more than a few seconds. One alternative is to use replication to a second system, which you can take offline periodically and then back up (the same strategy is often used with databases).

Conclusion

Considering how popular "big data" is becoming, it's no surprise that backup solutions and strategies are playing catch-up. For NoSQL solutions, creating backups while keeping the system running can be a challenge, so planning for these situations now can save you a lot of grief later on. And, chances are, if you aren't using NoSQL yet, you will be, so it's best to get ahead of the curve. At some point, we will hopefully get good deduping capabilities built into existing backup programs, because things will just keep getting bigger. ■■■

INFO

[1] Amanda: *http://www.amanda.org/*
[2] Bacula: *http://www.bacula.org/*
[3] Opendedup: *http://opendedup.org/*