Chapter 6

Secure Backup

Problem Statement

Alice considers her data very important. She has been around long enough to experience the painful loss of files due to arbitrary failures of software and hardware. The data on Alice's machine is of a very sensitive nature. She is very good at physically securing her machine and protecting her data while it is in her possession, but how does she back up her data in such a way that the backups are reliable and also secure?

Threat model

The adversary in this scenario is a user who manages to get read access to Alice's backup tapes. It should be impossible for him to learn anything about the data that is stored there. In addition, it should be impossible for an adversary to destroy the keys that are used to protect the backup tapes.

6.1 Secure Backups

When I give talks about computer security, viruses, worms, Trojan horses, and other threats, I'm often asked what the state of the art in defense mechanisms is. My reply is always "back up, back up, back up." If you have never lost any data due to some kind of failure that wasn't your fault, you have probably not been using computers for very long.

If you ask me what the easiest way to steal information from a highly secure site is, I will probably not suggest trying to exploit a misconfiguration in the firewall and subverting the perimeter protection to get an account inside and then using that account to break into a protected database. A much easier way

is to bribe the truck driver, who transfers the backup tapes from the building to a physically secure site, to look away for a couple of hours while you copy the tapes. This attack will only cost you a few hundred dollars; you may even be able to pull it off for a six-pack of beer.

Backup is one of the most overlooked processes when it comes to site security. However, backup is crucial. Backup is important for recovering from loss due to accidental or malicious failure. You would be hard-pressed to find a person or organization that hasn't had to restore from a backup at some point. When faced with data loss or corruption, the backup archive is one of the most appreciated and loved objects in the entire universe.

What is interesting is that even though backup tapes, by definition, contain data that is just as sensitive as the data being backed up, they rarely receive the same protection as the original data itself. Why is that? Well, the purpose of backup is to recover after some kind of a problem. So, if encrypted data is backed up in its encrypted form, then what happens if the unfortunate event that led to the loss of data also results in a loss of the keys? Encrypted backups without the keys are about as useful as a wad of cash when you are stranded on a deserted island. It seems like they should be worth something, but trying to use them proves futile. Even if you were to store the money away until you were rescued, by the time that happened, inflation would make the wad of cash practically worthless, but here, the analogy kind of breaks down.

6.2 Physical Security

One approach to secure backup is to physically protect backups. If you are an individual user, then you can purchase an external Jaz drive, or a PC card with FLASH memory, copy your sensitive files to the external device, keep it in your possession at all times until you get home, and then bury it ten feet deep in your back yard. Make sure to mark the spot carefully, and to put a mean dog in the yard, preferably one that does not like to dig.

If you are an organization, you could implement a process whereby backups are done under the supervision of security personnel, and the tapes are physically transported to a safe location.

I don't like relying on physical security for several reasons. First of all, it is difficult to find security personnel who are completely trustworthy. Most security compromises are initiated by insiders, and there are few physical security

types who are paid as much as the value of your data. Put another way, you'd be crazy to spend more on your security personnel than your data is worth. So, you are potentially vulnerable to bribery of your security personnel.

Physical security is not a bad idea, but I do not recommend relying on it exclusively to protect your data. Instead, couple it with software protection (encryption and authentication).

6.3 Backup over a Network

Backup data is vulnerable to attack at several points. If you are backing up your data onto a physical device such as a Jaz drive, then you do not need to worry about somebody sniffing on the physical connection between your computer and the drive. However, most backup techniques today involve transferring data over a network. It doesn't make sense to use strong encryption in your backups and good key recovery mechanisms, if you transfer files to a remote backup server in the clear.

The right way to back up files remotely is to perform all of the compression and encryption (in that order!) locally, and then to transfer the backups to the remote site for storage. The reason to compress before encrypting is that encrypted data contains very little redundancy, and so compression of ciphertext is not very effective. Many remote storage facilities further encrypt the data. While the encryption of the data on your local machine protects you against network attacks and from the storage server, the further encryption at the remote server is intended to protect your data from compromise of the server in the case where you have a poorly chosen passphrase. The super-encryption (encrypting encrypted data) at the remote site is a great marketing gimmick by many of the backup storage vendors, but it doesn't really buy you much because you should protect it with good keys in the first place. Furthermore, you are now running the risk of not only the loss of your passphrase, but loss of the key used by the backup server.

Another issue in the remote backup process is user authentication. If you back up your files over a network to a centralized server, make sure that the server does proper user authentication. If it does not, then even though the information on that server is unreadable, assuming it is properly encrypted, there may be nothing preventing another user from corrupting or destroying your backups.

Many remote backup facilities allow for an automatic unattended backup to be scheduled. That means that users can tell the system to make a backup in the middle of the night of files that have changed. Of course, the whole purpose of this is to perform a backup while the user is sleeping. It is unlikely that the user will want to wake up each night and enter the passphrase to derive the key for the backup. So, these systems require that the key be available to the program whenever it needs it. To accomplish this, the key must be in memory on the computer. In practice, many vendors keep the user key on disk somewhere. In either case, the key is vulnerable. The most secure systems require a passphrase to be entered whenever a backup or restore is about to take place, and they erase the key from disk and memory as soon as the work is done. Unfortunately, this is rarely the way these products operate.

Another common "feature" of many remote backup products is that they give the user a choice of key lengths and algorithms. In several cases, products offer 40-bit DES, 56-bit DES, 3-DES, and Blowfish or CAST. Average users are about as qualified to pick the bit size of their keys as they are to set the correct refresh rate on their computer monitor. The difference is that when setting the refresh rate on a monitor, you get some feedback if you select stupid settings. With crypto, you just get an insecure system. When questioned, one vendor replied that 3-DES is too slow for some users and that 40-bit is included in the product for export reasons. Huh?!? I asked him if the 40-bit version and the 3-DES version shipped as different products, and he said no. Apparently, there are companies out there that think their product is exportable if they add weak crypto to it, in addition to the strong crypto.

6.4 Key Granularity

The most common technique for protecting backups is to encrypt files locally using a key derived from a passphrase. There are several commercial products that do this, as I will discuss shortly. One choice that needs to be made is how many keys to use. If you use one key to encrypt all of the files that are backed up, then loss or compromise of that key means loss or compromise of the entire archive. Breaking backups down into finer-grained keys is much more complicated and difficult to maintain. You could have a program with a database for controlling all the keys, but you had better back that database up very carefully. In the end, the problem reduces to protecting and backing up keys securely.

6.5 Backup Products

There are many commercial offerings that provide backup service. Here are some of the more interesting ones. Ultimately, you should try several of them out for yourself before deciding on one. A great list of companies that provide backup can be found at `http://uk.dir.yahoo.com/Business_and_Economy/Companies/Computers/Business_to_Business/Services/Backup/`. Many of these have a security component to them. You can use the information in this chapter to help evaluate the level of security that they offer. Keep in mind that just because a company says that they use triple DES to encrypt does not mean that their product is secure. Here is a list of questions to ask yourself before choosing a product:

- Does this product compress and encrypt locally before transmission?
- What encryption algorithms are used?
- Do they also perform data authentication? If so, what authentication algorithm is used?
- How are keys derived?
- Is there a secure channel between my computer and the server?
- Are file names protected, as well?
- What is the key granularity?
- Does the server super-encrypt?
- How easy is it to restore files?
- Is there user authentication for storage and recovery?
- What is the user interface for backup and recovery?
- Is there any reason I should trust this vendor?
- Do they use well-known published algorithms and protocols, or their own proprietary ones?
- How are the keys stored?

Keep in mind that almost all network backup systems require you to install client software. Installing client software over the Internet and even from a CD-ROM is a sensitive operation. It is a point at which an adversary could

install a virus, or worm, or Trojan horse on your machine. By installing a native application, in a sense, you are completely trusting the distributor with your computer. While a product may claim to encrypt before sending a file, you have no guarantee that the product isn't also shipping the key in some covert manner (say, encrypted with a key that only the vendor knows) back to the vendor.

Also, even if the software vendor is totally reliable, there is no guarantee that their site has not been hacked. There are public domain tools for inserting a virus or any other malicious code into an existing application. A favorite is a program called *infect*, which asks for a filename to infect and the name of a malicious program, and then installs the malicious program in the target executable. I saw a demo of this. It's pretty scary.

With that in mind, here is a summary of the products. As you can see, the amount of information I was able to collect about these products from their literature and from contacting the companies for technical explanations varies widely.

6.5.1 @backup

This product has one of the nicer user interfaces. You can simply right-click on a file and you have the option to back it up. You can also restore different versions of a file based on the date it was backed up. All of the files are encrypted using a single key derived from a user passphrase. There is no claim that the product provides authentication in addition to encryption. The product encrypts files locally and then super-encrypts them on the server with the vendor's key. The product is pricier than most, costing $99 per year for 100 megabytes at the time of this writing. The URL is `http://www.backup.com/`.

6.5.2 BitSTOR

This product has a similar interface to the previous one, that is, using an Explorer-like interface to determine which files to back up. There is an automated unattended backup facility available. The user has the option of using DES, 3-DES, or Blowfish. The encryption key is derived from a user passphrase, and there appears to be no data authentication. One nice feature is that there is a separate user authentication stage for storage and recovery, so users need to remember a passphrase for the key and a user password to authenticate. The product also sets up an encrypted channel to the server before any communication takes place. In addition to encrypting the data,

this product also encrypts the file names for additional privacy. The URL is http://www.BitSTOR.com.

6.5.3 Secure Backup Systems

It is not clear how this product encrypts or how keys are chosen. The program compresses and then encrypts data that needs to be backed up, and it is stored in a physically protected offsite vault. The product automatically scans for changed files within a selected area of the file system and marks data for backup.

6.5.4 BackJack

This is a product that was designed specifically for the Macintosh. However, there is nothing in the protocol that is specific to the Mac. It provides for remote backup and restore over an insecure network. Users receive the first 100 megabytes for free, as of this writing. The product uses a passphrase-derived key and performs CAST encryption with a 128-bit key and MD5 for authentication. The URL is http://www.backjack.com/.

6.5.5 Datalock

This product is another remote backup-and-restore product. Files are compressed and encrypted locally with 3-DES using a passphrase-derived key. There is a facility for scheduling unattended backups. The URL is http://www.datalock.com/.

6.5.6 NetMass SystemSafe

In this remote backup-and-restore system, users are given a choice of 40- or 56-bit DES, or 3-DES, and keys are derived from a passphrase. There is a nice graphical user interface for administering the backups. An interesting feature of this system is that it can back up partial files by only backing up the disk blocks that have changed since the last backup. The URL is http://www.systemrestore.com/.

6.5.7 Saf-T-Net

This remote backup-and-restore program performs user authentication via an ID and a PIN. The product tries to obtain security by obscurity (deplored by the security community). It implements a proprietary communication protocol with variable-length packets and proprietary compression to achieve “security.” It has an additional property of performing virus checking on the client

side before copying the backup files to the server. There is no mention of encryption or keys—proprietary compression seems to be used for that purpose. This goes against all of the conventional wisdom in the security community. The URL is `http://www.trgcomm.com/`.

6.5.8 Safeguard Interactive

This product uses a user-supplied passphrase to derive a key used for DES encryption. Files are backed up and restored over a network. The URL is `http://www.sgii.com/`.

6.5.9 Veritas Telebackup

In this product, authentication of files is done using a cyclic redundancy check (CRC) instead of a cryptographic hash. This is not good practice. There is an authentication code stored in the client software to validate a session. This is also the wrong way to do it. Not only are sessions not encrypted, but anyone who ever gets access to the software can then spoof a session. There are a lot of obscurity games in this product such as variable-length packets, proprietary compression in place of encryption, and randomized storage of files on the server. None of these things would hold up against a serious attacker. Why not just encrypt and MAC the data, authenticate users, and derive 3-DES keys from passphrases? Everybody else seems to understand this to some degree. If you are curious, the URL is `http://www.veritas.com/us/products/telebackup/`.

6.6 Deleting Backups

While data backup provides a convenient way to recover from crashes and other losses of data, it comes at the cost of long-term persistence of the data. Imagine that you have a file that contains all of your old e-mail. At some point, you realize that having such a file implicates you in several “situations” that you would rather forget. Deleting your mail file is not enough. You have to delete all of the backup copies. If you did a very good job backing up your files, then there are many, many copies of the file at all different stages on all sorts of backup servers. Hopefully all of them are encrypted, but if you used a weak cipher such as 56-bit DES, which was believed to be secure several years ago, then that won’t be very useful. Even if they are encrypted, you may discover

that your backup software kept a copy of the key, which was the same key that you used to encrypt all of your backups on the local disk, and that you were hit by a virus that targets that backup system and copies the keys to remote locations. Yikes!

Boneh and Lipton [19] describe a revocable backup system. In their system, all data is encrypted by short-lived keys that expire at intervals defined by the user. A master key is used to encrypt all of the keys in the system. To make a backup of a file useless, all a user has to do is erase the key that was used for that file, and all of the previous versions of the backed-up file are rendered useless. In practice, the master key could be derived from the user-defined passphrase that is used in existing commercial systems, and the details of the revocable backup scheme could be hidden from the users.

6.7 Case Study

In this case study, I will walk you through the proper design of a remote backup system. Keep in mind our trust model—the local environment is trusted; the network is not, and neither is the remote server.

Assume that there is a secure way to obtain a client-side program. While this is a leap of faith, we have to start somewhere. Perhaps the client backup program has a well-known hash, and you are able to verify it on the client end. Anyway, if you cannot obtain a secure version of your security software, you are in big trouble.

So, what is the software that you are running locally? Ideally it is a cryptographic file system like the ones described in Chapter 4. If that were the case, then you could simply ship out the encrypted versions of files and store those remotely.

There are several reasons why this is not practical. Cryptographic file systems (CFSs) require some understanding on the part of the user that he is using a CFS. Also, the installation may not be trivial. Not all users are sophisticated enough to manage this. Furthermore, the user may be running applications that do not let him control where files are stored, so there may be no way to mount those files in a CFS. Thus, in this case study, we focus on a backup system that is retrofitted to a commonly used environment, such as a Windows PC.

6.7.1 The Client Software

In my view of the ideal remote backup system, a user first starts a session, which is an interaction with the software for the purposes of backup or restore. When the user starts a session, he is prompted for a passphrase. He then selects whether or not this session is a backup or a restore. If it is a backup, then the system does some proactive checking and makes sure that the passphrase has enough entropy. One good way to accomplish this is to show a progress bar and require the user to keep entering characters until the progress bar is full. A sensible algorithm is then used to derive two 128-bit keys from the passphrase. The first is for authentication, and the second is for encryption. In practice, the user should probably use the same passphrase for all sessions, otherwise he is likely to forget it or write it down somewhere.

The client software would ideally resemble a nice graphical file manager. Perhaps it could be identical in look and feel to Windows Explorer, with folders and icons for files. In fact, a very good program would simply add functionality to the existing Windows Explorer. The user presses the shift and control keys and uses the mouse to select which files to back up, or alternatively, picks from a previously saved list of files. Next, the user activates the backup by pressing a button or selecting from a menu. For security reasons, unattended backups are not allowed.

At this point, the software kicks in. First, a *bundle* is created. Each file is compressed and added to the bundle. In practice, this could be the same as a zip archive or a UNIX `tar.gz` file. Then, the authentication key is used to compute the HMAC (see Chapter 8) of the bundle, and the output is added to the bundle. Finally, the bundle is encrypted with the encryption key using a strong block cipher, such as triple DES or AES. The bundle is then tagged with the user name, the address of the user's machine, and a time and date, and is sent over to the untrusted remote backup server. The remote server then stores the bundle, indexed by the tags. One nice thing about this way of doing things is that the file system structure and the file names are hidden from the remote server and from anyone listening in on the network.

If the session is a *restore,* then the user is prompted to pick a date. A list of all of the previous backup dates is downloaded from the server and shown, and the user selects which date he wishes to restore. The software imports the corresponding bundle from the server and decrypts it using the key derived from the passphrase. The authentication is then checked, and if it verifies correctly, the restore proceeds. Next, a Windows Explorer view of all of the

restored files is presented, anchored at a new root directory. For example, the old file system view is mounted at `C:\restore\old_root`. The user can preview all of the files in their restored format and decide to accept or reject the restore. If it is accepted, then all of the files are restored in the actual file system. The user can also select to restore on a per-file basis as opposed to taking the whole bundle.

One interesting feature of the scheme presented here is that there need not be any user authentication for a restore session. The servers can make all of the bundles available to the world. The strong encryption and authentication properties make them tamper evident and opaque to anyone who cannot obtain a user passphrase or break the authentication and encryption functions.

However, it is desirable to have some user authentication when a user performs a backup. Otherwise, attackers could fill up the disks on the servers with anything they wanted. Users should be strongly advised not to use their data backup passphrase to authenticate to the remote backup server.

6.7.2 Incremental Backups

In a typical backup scenario, the user selects a *set* of files in the file system to back up. The set is often given a name or an icon, and there is an easy mechanism for the user to execute a backup of that set of files. It is inefficient to back up all of the files in a particular set every time. A common technique for avoiding this is to perform a full backup periodically, in which all of the files are copied. Then, whenever a backup is needed in between full backups, an incremental backup is done. An incremental backup consists of copying only those files that have changed since the last backup. To accomplish this, a local database is maintained containing filenames and modification times for all files that have been backed up. When it is time for an incremental backup, the software checks this database to see which files in the file system have a more recent modification time than is shown in the database, and these files are backed up.

To restore incremental backups, the system simply restores the most recent full backup and then restores each of the subsequent incremental backups in order of least recent first. Thus, files get restored to the view of the file system at the last incremental backup. It is not necessary to back up the incremental backup database, as long as the order of incremental backups is maintained on the remote backup server. If the database gets trashed, then the next backup must be a full one.

There are, however, some security considerations when doing incremental backups. Assuming that there is an adversary out to get you, there is a basic attack that could wreak havoc with your backups. If an attacker can change the modification times in the database, he can set the time ahead, and the system will not back up files, even though they have changed. In fact, in most deployed systems, an attacker could set the modification times of all files in the database to some time far into the future, and the software would probably not detect it.

The defense against this attack is straightforward. When I discussed performing backups, I already described two keys: an authentication key and an encryption key that are derived from the user passphrase and available in memory at the time of the backup. A secure remote backup system should use the authentication key to compute a MAC on the incremental backup database after every legitimate change. The MAC can be stored together with the database. The cryptographic properties of the MAC are such that nobody can modify the file or the MAC in a way that modifications to either will not be detected. Of course, it is important that the MAC be verified before every incremental backup. Again, keep in mind that attacking the database only disrupts the backup process, not the restore process. The database is not used for restoring files.

6.8 Further Reading

For more information on the material in this chapter, check out the following resources.

Articles

D. Boneh and R. Lipton. A Revocable Backup System. *USENIX Security Conference VI*, pages 91–96, 1996.

Web Sites

`http://uk.dir.yahoo.com/Business_and_Economy/Companies/Computers/Business_to_Business/Services/Backup` A great resource for finding online backup systems.

Here are the Web addresses of the companies providing backup systems that are discussed in this chapter:

http://www.backup.com/
http://www.BitSTOR.com/
http://www.backjack.com/
http://www.datalock.com/
http://www.systemrestore.com/
http://www.trgcomm.com/
http://www.sgii.com/
http://www.veritas.com/us/products/telebackup/