A Secure Remote Database Backup System

JOSÉ VICENTE AGUIRRE[1], RAFAEL ÁLVAREZ[2], JOSÉ NOGUERA[3] and ANTONIO ZAMORA[4]
Departamento de Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante
Campus de Sant Vicent del Raspeig, Ap. Correos 99, E-03080, Alicante
SPAIN

Abstract: - We propose a secure remote database backup system. This application allows for secure session establishment using public key cryptography, data backup using two kinds of algorithms, compression and encryption of backups and secure storage on a remote server. The security of the application is adequate even for sensible business data.

Keywords: - data security, remote database backup, redundant storage.

1 Introduction

Surprisingly, about 90% of small businesses do not perform security backups or they backup data incorrectly suffering the corresponding loss of productivity, income or clients. The most common problems are accidental data erasure, file overwriting, new software installation, robbery, fire or natural disasters, hacker attacks, spyware, etc.

Remote and encrypted backups guarantee that, in case a disaster occurs, be it at the physical location of the computers or inside the computers themselves, data is available immediately.

The TreeKeeper application performs a backup of a server, workstation or laptop (Windows XP, 2000 and 2003 environments) using the Internet, intranet or dial-up as a transmission channel, storing the backup on a remote server located at a different physical location but connected to the Internet or any other TCP/IP network. All data is sent encrypted across TCP/IP networks fully guaranteeing data confidentiality and application functionality.

There are several reasons why this tool can be useful for several types of businesses. In the first place, it must be considered that data is generally a business' most valuable asset because they are required for everyday activity. Secondly, the law requires that high level data must be conveniently protected and a copy securely stored offsite.

In this paper we analyze and describe the security strategies and techniques followed during the development of TreeKeeper.

2 Description

The application performs unattended backup copies, at a user programmed time, which are also encrypted to guarantee data confidentiality. On the other hand, it is also possible that backups have local or remote second copies. Backups are sent to the remote server using a secure channel.

Having security always in mind, and pursuing the ease of usage, system security is guaranteed by two keys: the master key, that allows access to the system; and the current key, used to encrypt the backup files in such a way that all backups are encrypted with the same key until it is changed.

The master key is required:

- Each time the user interface is started.
- Each time that the operating system is started.
- Each time the application is started by the user using the supplied start-up tool.

In order to send backups to the remote server, or receive them from the server, it is necessary to have a FTP server or, at least, a proprietary storage server.

Regarding the backup process, the application supports two types of backups: full or file level incremental. When a full backup of a set of files is performed, all referenced files are always copied (even if they have not changed since the last backup). In the case of an incremental backup, only changed files since the last backup process are copied. In the domain of our application, a backup is defined as the set of files to be copied and the corresponding backup policies determining periodicity and type of backup (full or incremental) performed. All backup copies have a default backup policy, called instant policy, which cannot be deleted and can be used to perform backups at any time. Also, every backup policy configuration is encrypted using the same algorithm chosen to encrypt the backup. All backups are compressed first using the ZIP algorithm [6].

Backup policies can be paused indefinitely either by user request or automatically (as a consequence of an unrecoverable error during backup operation).

A finished backup is the successful result of a backup policy (since a backup policy determines the execution periodicity, a single backup policy may have multiple finished backups as time progresses). It is possible to have up to three physical copies of a single finished backup:

- *Local copy:* finished backup stored locally and main source for restoration.
- *Safety copy:* second (safety) backup of the finished backup.
- *Remote copy:* finished backup stored on a remote server.

The recovery of local copies is transparent to the user: after choosing the recovery option desired, the system automatically determines the correct key to decrypt and restores the data. Recovery can be performed at two levels:

- *File level:* only certain specific files are restored.
- *Backup level:* the complete backup containing all files is restored.

These options are available regardless of the type of backup (full or incremental). It must also be considered that if the backup is incremental, it may require increments not available locally.

It is possible to delete backups and their policies. If a policy is deleted, the corresponding finished backups are assigned to the instant backup policy.

A backup deletion destroys all files linked to that backup. It is possible to either perform a virtual local backup deletion (to free local hard disk space and still being capable of restoring data from the remote server or a second safety backup) or a complete deletion, eliminating all physical copies of that backups and associated links (in the case of an incremental backup).

Backups are executed in strict FIFO order, under a temporal sorting criterion. Only one backup can be in execution at each single time instant since multiple simultaneous disk accesses would cause a considerable slowdown in machines responsiveness. If a certain backup is delayed then it is executed as soon as possible.

Regarding data interchange with the remote server, the number of upload and download connections is configurable, being possible to cancel any operation by user request or connection error. Data is always sent resuming any previous connection in order to save bandwidth in the case of connection losses. Data transfer operations are simultaneous since Windows operating system limits each socket speed so, in this case, there are no performance problems unless the number of operations is too high.

Due to the fact that the system has been designed to be executed in the background, the interaction of the user interface with the related services is performed in a polling fashion or periodic checking, so some operations require refreshing so that they are effective immediately. Nevertheless, the time delay between each refresh is about 5 seconds (depending on system load) for backups and 1 minute for data upload.

In addition, a system log has also been included, informing the user about all system events (success, error or warning).

3 Application Analysis

In this section the protocols and strategies employed in the application are described in detail.

3.1 General application architecture

The application is comprised of four highly independent fundamental blocks, that interact with each other by the corresponding interfaces:

- *Graphical interface:* allows backup administration.
- *Oracle service:* process in charge of centralizing data access and providing data transparency.
- *Planning service:* process in charge of executing backup policies.
- *Messenger service:* process that manages backup exchange (sending and receiving) with the remote server.

The chosen operating system has been Microsoft Windows XP, although it will also work under Windows 2000 or higher that support the .Net Framework version 1.1 or higher.

Due to the fact that the system allows performing unattended backups, it is paramount that processes can be executed in the background. For this purpose the system employs Windows services (formerly known as NT services) that execute in the background without requiring any user interaction. This functionality is not available in versions prior to Windows XP that are not of the NT family.

The communication between different services and the user interface is done by the means of a remote object (served by the Oracle service), that has all the functionality associated to traditional servers but with the advantage that it behaves like an object inside the application. This remote object installs at the local host on a fixed port.

Since the remote object is accessible through the network, it is necessary to incorporate an authentication system to grant transaction execution permissions based on session identifiers, list maintained by the Oracle service itself.

The Oracle service is the main part of the application since it grants access to all required data. Therefore, to access any data it is necessary to obtain previous positive authentication using the master key. So, even before starting up the services or the GUI, the systems requests the master key from the user in order to provide authentication. From this instant, each running service will have a session key to connect to the Oracle service. The session key is valid until the machine is restarted. Starting services up (except the Oracle) without this previous step will cause the error to be logged in the operating system's error log and the services to be stopped.

The authentication system is based on the system master key that is stored in an encrypted file with AES using 256 bits of key size. The information related to the remote server is also stored in a different file encrypted in the same way, in order to prevent unauthorized information access.

The secure communication channels with the remote server can be done in two ways: using the SSL protocol with an sFTP server, or using a secure communication scheme (Needham-Schroeder, see [1] for more information), consisting of session key exchange and encrypting all successive data. This is a lighter scheme more suitable for mobile devices.

3.2 Password System

The systems consist of three types of keys: the master key, the current keys and session keys.

Session keys are all keys used each time a communication is established between two parties in an insecure network. The keys are exchanged by the cited key exchange algorithm, are cryptographically random and last until the session is terminated. In the case that any of these keys was compromised, it would only affect that session. The chance of this happening is very low considering the protocol employed for its distribution [1].

Current keys are the keys used to encrypt all finished backups and change during time. The validity period for these keys is user defined and for each period the system generates a cryptographically random key that will be employed to cipher the backups performed during that period. The system also allows the user to define the key for that period, which can be useful if the user wants to recover the backups manually. If any of these keys was compromised, it would only affect the validity period for that key. In this case, human factor causes that keys are somewhat easier to compromise than session keys, for that purpose the system includes a filter for non recommendable keys [5], although a dictionary or brute force attack must not be underestimated. This is the reason that the SHA512 algorithm is used to generate the real bit sequence employed as key. In order to decrypt the backups it also necessary to maintain a history of all current keys employed. Such history file is a weak link since and must be well protect, since if it was compromised all of the copies performed by that user could be decrypted. All current keys are of 512 bits in length, if the cryptosystem chosen has a smaller key size then only the required most significant bits are used.

The master key is necessary to encrypt the history of current keys file. This key must be defined and remembered by the user, who must also be responsible of its security. The master key is not stored anywhere, it is determined to be correct if the history file is decrypted successfully. If the master key was compromised, all of the current keys would be compromised too, so the security of the master key is extremely important.

This system is considered safe for high security profile installations as long as the master key is kept secret.

3.3 Backup Integrity

The system guarantees the integrity of the backups storing a 512 bits hash of all backups performed after being compressed and encrypted. For that purpose, the SHA512 algorithm is used. This hash function is employed to avoid data corruption or unauthorized manipulation.

Additionally, an independent hash of each file contained in the backup is also stored in order to detect changes between incremental backups and to guarantee that the restored files are identical to the original files.

4 Conclusions

In summary, TreeKeeper is a robust application for the creation and remote storage of database backups. It also shows a high level of security employing well known cryptographic algorithms and standards. It has been demonstrated that the application has enough security measures to reduce the risk of compromised keys, may it be preventing the usage of weak keys or limiting the damage caused by exposed keys. Finally, data integrity is guaranteed storing and checking a hash of the data on every operation where data corruption could occur.

A possible improvement would be guaranteeing non repudiation from the server so that a user would have legal means to justify the loss of data by the server. The performance with big files or in the case of backups consisting of lots of small files could also be improved. Finally, a way to perform bit level incremental backups could also be implemented in addition to the file level incremental backup already available. These changes should have no effect on the necessary security techniques analyzed in this paper.

References:

[1] Needham, R., Schroeder, M. Using encryption for authentication in large networks of computers, *Communications of the ACM*, 21(12):993-999, 1978.

[2] Stallings, W. *Cryptography and Network Security. Principles and Practice. Third Edition.* Prentice Hall. New Jersey. 2003.

[3] Handley, M., Aciri, H., Schulzrinne, E., Schooler A., Rosenberg J. Session Initiation Protocol (SIP). *The Internet Society* 1999.

[4] Menezes A., van Oorschot P., Vanstone, S., Handbook of Applied Cryptography. CRC Press. Florida. 2001.

[5] Password Management Guideline, *Department of Defense*, CSC-STD-002-85.

[6] Welch, T.A. A Technique for High Performance Data Compression, IEEE Computer, Vol. 17, No. 6, 1984, pp. 8-19.