Technology Whitepaper

Updated November 8, 2006

The Backup system has been designed to be a secure, reliable, efficient, scalable, modular, and portable data backup solution. An overview of each of the following components will be given: data encryption, revision management, pass phrase recovery, client/server protocol, data repository, and account management.

Executive Summary

Each file is compressed and encrypted using the AES-256 standard using a user-provided pass phrase as the key. The encrypted information is sent across an authenticated and secure (SSL/TLS) Internet connection to the server, where it is stored encrypted on the RAID-6 array.

Data Encryption

The information is encrypted by the client program using the AES-256 symmetric encryption algorithm. The 256-bit encryption key (as well as another 256-bit HMAC key) is generated by running the PBKDF2 algorithm on the pass phrase as described in RFC 2898.

Each file is divided into 2K blocks. Each block is compressed using zlib deflate and is encrypted using AES-256 in CTR mode (each block uses a different crypto-random nonce), and an HMAC (using SHA-256) is appended (to guarantee integrity upon restore). The data is fully compressed and encrypted before it ever enters the network. Encrypting the data on the client is more secure, and it makes the server more efficient and scalable.

Filenames are currently not encrypted. Encryption of file and directory names is a planned future enhancement, and can be done by upgrading the client software – no changes need to be made by the server.

Revision Management

The use of compression and encryption precludes the possibility of performing file delta calculation at the server. Performing the delta calculation at the client also increases server scalability and efficiency.

As each block is stored on the server the client stores a 64-bit CRC (two 32-bit CRCs generated from different polynomials) associated with that block. During the next backup if a file has been changed (its modified date/time stamp has changed or its size has changed) then the client will compare the new 64-bit CRC with the stored CRC. If the CRC has changed the client will upload the new compressed and encrypted block. Otherwise the client will tell the server that block has not changed.

The CRC block fingerprints and other data are protected by transactions such that if the backup of a file fails, all 64-bit CRCs and other information are rolled back to the consistent state.

Pass Phrase Recovery

The pass phrase is central to the system's security. A strong pass phrase is vital for sufficient encryption strength. However, a strong pass phrase can be hard to remember, and all data would be worthless without the exact pass phrase. Thus, a secure pass phrase recovery system has been implemented.

When a pass phrase is created or changed the user answers several security questions. The answers to the questions generate an encryption key, which is used to encrypt the pass phrase. A new random 256-bit encryption key is generated, and the data is encrypted again. The 256-bit encryption key is then encrypted with a 3072-bit RSA public key. The associated private key is encrypted and secured by an extremely secure pass phrase only known to master technicians.

Pass phrases are stored in this dually encrypted fashion on the server. To recover a pass phrase the client program generates a random 3072-bit RSA key and sends the public key in the request file to the server (ensuring only the client that generated the recovery request can attempt to recover the pass phrase).

A master technician decrypts the outer layer and re-encrypts it with a new random symmetric key, encrypting the new random symmetric key with the public RSA key in the recover request. The client program downloads the response and decrypts the outer layer. The user must then correctly answer the security questions, thus allowing the inner layer to be decrypted and the plain text of the pass phrase to be recovered.

The pass phrase stored on the server is thus secure, requiring the cooperation of both a master technician and the end user. Only the end user ever sees the recovered pass phrase.

Client / Server Protocol

Each end user is given an account username and password (which can be changed). The client connects to the server via a TLS (SSL) connection (providing confidentiality and integrity), and requires the server's certificate to be issued by the eSecure Repository root certificate authority (to prevent spoofed servers from stealing login credentials).

The client authenticates to the server with its username and password. At this point the server may redirect the user to a different server and port. This greatly increases scalability, as all clients point to the same login server, but can be redirected to their data server according to need. Data servers can be added on demand, and an account's data can be moved to larger servers as the account grows. The end user is not aware of this complexity and never needs to change anything.

The communication protocol itself is an endian-independent, flexible protocol designed to support changes without breaking backwards compatibility.

Data Repository

All data is stored as files within the server's file system. The server is portable and runs either in Win32 or Linux. Servers currently run on SuSE 9.3 Linux using the ReiserFS 3.0 file system with the tea hash function. ReiserFS is a modern journaling file system supporting multi-terabyte partitions, 64-bit file sizes, and millions of files per directory, making it preferable over NTFS. RAID-6 is used for all repository partitions.

Each account is assigned a subdirectory and contains subdirectories for root backup folders. Each root backup folder contains the following subdirectories:

- data: Current versions of files
- meta: Historical versions of files
- deldata: Deleted versions of files
- delmeta: Deleted, historical files
- index: mirrors the directory structure so the directory list can be generated quickly

The current version of the file always stores the complete file (encrypted and compressed). Historical versions store data blocks that differ from the next (more recent) version. Thus, to restore the 5th version of the file you apply the deltas from the previous 4 versions and then apply the 5th delta. This is done as the file is downloaded and is very efficient. Also, this method makes uploading new versions efficient, as all previous versions need not be changed.

When the client detects that a file has been deleted it notifies the server during the next backup. The server annotates the filename with the deleted date/time and moves it to the deleted data area. The client program will enumerate and destroy old deleted data once a week.

An end user can use the file manager to destroy data. When data is destroyed it is moved to a parallel repository designed to hold the "destroyed" data. Destroyed data is held for an additional 30 days, in case the destruction of data was unintentional.

All actions in the repository are transactional so that the system is always in a consistent state. A transaction log is kept on the disk such that if the server ever loses power the transaction will be rolled back upon server startup and the system will be restored to a consistent state. Transactions are also automatically rolled back if a network connection times out or some other error occurs.

The repository uses "meta data" objects to track disk usage. There is one meta data object per directory, and it tracks how much data is contained within that directory and all of its subdirectories. These meta data objects are updated in real time in a transactional manner. This allows the server to provide disk usage information to the client program (or billing process) in a very efficient manner.

Because all repository data is stored as files within the native file system, an account's data can be managed easily using the native operating system's utilities. Additionally, existing technologies and utilities to mirror file systems (such as rsync) can be used to provide additional protection against data loss.

Account Management

All account information is stored in a PostgreSQL database. The login server and the data server must connect to the same database, or replication must be employed to keep the databases consistent. The database also contains billing information and a detailed audit trail.