Using Free Web Storage for Data Backup*

Avishay Traeger, Nikolai Joukov, Josef Sipek, and Erez Zadok

Stony Brook University
Computer Science Department
Stony Brook, NY 11794-4400
{atraeger,kolya,jsipek,ezk}@cs.sunysb.edu

ABSTRACT

Backing up important data is crucial. A variety of causes can lead to data loss, such as disk failures, administration errors, virus infiltration, theft, and physical damage to equipment. Users and businesses have important information that is difficult to replace, such as financial records and contacts. Reliable backups are crucial because some data cannot be replaced, while recreating other data can be expensive in terms of time and money. We propose two methods which leverage various types of free Web storage to provide simple, reliable, and free backup solutions.

The first method is based on the storage of data in the caches of Internet search engines. We have developed CrawlBackup, a tool which prepares and provides the data for Web crawlers and can then restore the data from the Internet even if all the data on the original computer is unavailable. The second method, called MailBackup, stores redundant copies of the important data in the mailboxes of Internet mail services. We have successfully used these backup systems since the middle of 2005. In this paper we discuss and compare these methods, their feasibility of deployment, their security, and their flexibility.

Categories and Subject Descriptors

E.5.a [**Data**]: Files—*Backup/recovery*

General Terms

Reliability, Security, Design

Keywords

Backup, Web services

*This work was partially made possible by NSF CAREER EIA-0133589 and CCR-0310493 awards and HP/Intel gifts numbers 87128 and 88415.1.

StorageSS'06, October 30, 2006, Alexandria, Virginia, USA.

1. INTRODUCTION

Users and businesses have important information that is difficult to replace, such as financial records and contacts. It is estimated that on average, just 20Mb of business data takes 30 hours to recreate and is worth $100,000 [1].

Redundancy is crucial for backing up important data [3]. To improve the safety of data further, redundant copies should be kept in locations which are as physically (and logically) independent of each other as possible. This is often difficult to achieve for the average home user or small corporation, whose computing resources generally reside in one location. Backup independence is difficult to achieve even for users in larger corporations or universities who have access to resources in several locations; these resources are likely to be on the same network and vulnerable to the same viral infections. Utilizing free services found on the Internet is ideal in this respect, because the data will reside on several distinct systems.

Free data hosting has been available for some time now, mostly in the form of Web hosting, Web-based email, and even picture and video storage. However, only recently has the amount of storage been enough to be considered a viable data backup solution. Of these storage options, email is the simplest and most convenient. Data that needs backing up can also be placed automatically on a Web page, where growing number of Web search engines will cache it. As with every security solution in general [6, 13], our solutions have advantages and disadvantages, discussed in Section 5. The key advantage of our proposed backup solutions is that they are free, which is frequently the winning factor in practice [7].

In addition to addressing concerns about the safety of the backed up data, a backup scheme should be simple and mostly automatic, support versioning, and provide the user with control over the backup frequency. Simplicity is important so that the backup process is reliable and easy to use. Automating the process helps to guarantee that backups will be made in regular intervals and that important data will not be lost because the user forgot to back it up manually. Versioning is a useful feature, because it is often the case that users need to refer to older records. In addition, it provides an extra level of safety in case a corrupted version of a file was backed up, leaving the user with no usable copies. Providing users with control over the backup frequency allows them to balance the amount of possible data loss with the amount of required storage. The Web-based backup techniques that we discuss in Sections 3 and 4 address each of these issues to different extents.

Backing up important files on systems that do not belong to the owner of the data raises privacy concerns. This is even more true when using Web search caching, since the data is viewable without a password. We will address this issue as it applies to each backup technique and discuss possible solutions. The key idea is encrypting the data before it is presented to the remote server for storage.

System	Storage	Max. Message Size
Gmail	2,757 MB	10 MB
GMX FreeMail	1,000 MB	20 MB
Hotmail	250 MB	10 MB
Seznam	2,000 MB	13 MB
Yahoo! Mail	1,000 MB	10 MB

Table 1: Total storage and maximum message sizes for some of the popular email services as of August 2006.

The rest of this paper is organized as follows. We discuss data and storage sizes in Section 2. We describe our backup solutions in Sections 3 and 4. Other considerations regarding these backup techniques are discussed in Section 5. We discuss prior work in Section 7, and conclude in Section 8.

2. QUANTIFYING DATA'S IMPORTANCE

Virtually all Internet services have some kind of limit on the maximum file size, as well as the overall storage quota. Interestingly enough, on a typical server most files are less than 256KB, and virtually all files are less than 10MB. Additionally, the distribution of file sizes has remained rather fixed over the past two decades [2, 14, 16]. Therefore, file sizes are not an issue when using Internet services to backup one's data when one considers the amount of storage offered.

Table 1 shows the amount of storage offered by several of the most popular email services as of August 2006. It is immediately clear that the amount of storage offered is large enough to back up a sizable portion of data. Additionally, one is not limited to just one account, so the usable storage is potentially even larger.

One does not need to backup all data with equal frequency. Files can be divided into four distinct classes depending on how frequently they need to be backed up [20].

- The first class consists of files important to the user, such as word processor documents, spreadsheets, and bank records. These files should be backed up frequently.
- The second class, which consumes 82–85% of storage, includes those files that have not been accessed in more than a month [2]. Files in this class can be compressed to recover space and do not need to be backed up as frequently.
- Third, multimedia files can be re-encoded. If a lossy compression algorithm is used, some of the original data will be lost. This is usually not a problem with file formats such as JPEG and MP3. These files can be backed up less frequently than others.
- The last class is made up of files that can be regenerated. Studies show that over 20% of all files are regenerable [15]. These files need not be backed up.

3. BACKUPS USING SEARCH ENGINE CACHING

Internet search engines' crawlers periodically crawl the Internet, indexing and storing some types of discovered resources in their caches. This allows accessing these resources even if they are no longer available at their original location. Some people benefited from these caches by recovering their Web sites after accidental deletion, hardware failures, or hacker attacks [4]. Therefore, the logical conclusion is to use these search engine caches to store redundant copies of important data.

We have developed *CrawlBackup*—a tool that provides data to the Internet crawlers, finds it in the Internet later and returns it back to the users. We have used this to back up important personal data since the middle of 2005.

3.1 Operation

CrawlBackup consists of two shell scripts: CrawlBackupCGI (a CGI script) and CrawlBackupRestore. The CrawlBackupCGI script is linked from some public Web page with the target directory or a file as a CGI script parameter. CrawlBackupCGI performs the following actions:

1. Checks its configuration file to verify that it is allowed to backup the requested data.
2. Checks if any of the files were modified since the last backup time based on the modification times of the files.
3. Creates a single compressed tar file of the requested data.
4. Optionally encrypts the data for privacy reasons, currently with a symmetric key.
5. Creates a uuencoded representation.
6. Generates the final page that consists of
    (a) The header with the backup identifier and backup date and time. The backup identifier is used to find the generated page later. It usually contains the name of the original directory that was backed up to allow manual searches for the data later on.
    (b) The data generated in the previous steps.
    (c) The CrawlBackupRestore script, which is included to restore the original data in case the script is corrupted together with the data.

The CrawlBackupRestore script is invoked manually and performs the following actions:

1. Finds the page generated by the CrawlBackupCGI by sending requests with the page identifier to a number of search engines. This step must be performed manually if the CrawlBackupRestore is unavailable.
2. Extracts the uuencoded part of the page and uudecodes it.
3. Optionally decrypts the data using they user's symmetric key.
4. Decompresses and untars the data.

3.2 Design Considerations

CrawlBackupCGI has a backup scope granularity ranging from individual files to large directory trees. Every link to the script corresponds to one such scope. Because CrawlBackupCGI is a CGI script, it is desirable to limit the execution time of its individual invocations. Therefore, we split large directories into smaller backup units when it is necessary.

The CrawlBackupCGI script operates using a configuration file to aid system administrators in maintaining backup rules for users. The target directory identifier is provided as a CGI parameter as a part of the URL for the script. CrawlBackupCGI then verifies that the requested directory is allowed to be backed up.

During the CrawlBackup design we experimented with different forms of data representation. In particular, CrawlBackup can convert arbitrary binary data into HTML pages, PDF documents, and

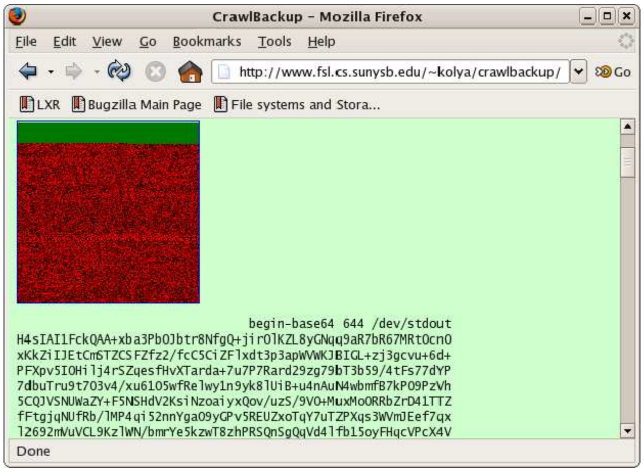

Figure 1: CrawlBackup-generated representation of data in the form of a bitmap image and as HTML-embedded text.

several image formats. For example, Figure 1 shows a CrawlBackup-generated page with the data represented as a bitmap image and HTML. However, we have found the HTML representation to be the most reliable. This is because data in other file formats may be corrupted due to compression or rescaling operations performed by search engines, whereas HTML data is most likely to be cached.

We have also experimented with representing the encrypted data as text. This was done firstly because we noticed that some search engines did not cache the uuencoded data. In addition, if search engines wanted to exclude backed up data from their caches, this would make it more difficult to recognize it. After optionally encrypting the data, we use a word list to transform the binary data into words. This is a similar idea to the S/KEY one-time password utility [5]. This increases the data size by roughly five times. It is also possible to use a grammar to create grammatically correct English sentences rather than choosing any word from a list [19].

4. BACKUPS USING FREE EMAIL STORAGE

Most users have emailed important data to themselves for safe-keeping on a mail server at one point or another. Some email services provide free storage, and automating the backup process using email can create a viable backup solution. We are currently developing a prototype of one such solution, called MailBackup.

4.1 Design Considerations

Creating an automated backup process based on free email services requires more extensive scripts than those we used for the search engine caching method. This is due to several factors:

1. Web email services do not provide uniform interfaces for sending, retrieving and managing email. Therefore, we were forced to look at alternative means of accessing email. The combination of POP and SMTP allows for access to the storage provided by the server. All the email services in Table 1 provide POP access. MailBackup can use any SMTP server, allowing it to work on any network connected to the Internet. Gmail, in addition to POP, provides a file-system-like interface. A Linux user-space file system, GmailFS [8], as well as a Windows Shell Namespace Extension [18], have been written to access it. This allows MailBackup to optionally use a higher level of abstraction and use ordinary system commands and tools, instead of using mail protocols.

2. There is a limit to the total amount of data as well as the maximum message size one can store using each account (see Section 2). This mandates an option similar to that of CrawlBackup to determine the maximum size of messages. Since email attachments are generally base-64 encoded, the total amount of storage required will increase.

3. Due to the need to use several accounts with potentially different password, the system maintains a $\langle$*user-name*, *password*, *server*$\rangle$ mapping and uses the appropriate data to authenticate and retrieve the status of a backed up chunk. Since data is still stored on remote servers, the data should be encrypted for added privacy.

4. Spam is not a problem if we apply simple filters which allow incoming emails from select addresses only. It is possible for spam to get by the filters only if the originating address is that which the system expects. If this address is kept confidential and is hard to guess, getting spam is unlikely.

The MailBackup scripts are more complex than the CrawlBackup scripts because of their "push" nature. The main difference between the two types of backup options is that backups to email storage are created by the user's request and push the data to the server, while backups to Web search caches are created when the crawler downloads the data.

The configuration of MailBackup is similar to that of CrawlBackup. However, in addition to the basic options such as maximum chunk size, a list of $\langle$*user-name*, *password*, *server*$\rangle$ mappings must be maintained, as well as the source email address and the SMTP server to use.

The major advantage of MailBackup is the fact that it can be run manually when the user requires immediate backup or can be set to automatically run at predetermined intervals. The files and directories to be backed up are configured in the same way as CrawlBackup.

There is a separate account manager that keeps track of various email accounts and passwords. Passwords are stored in encrypted form, and the user supplies one master password to unlock all accounts. The account manager must also keep track of quota usage, decide where to store new backup versions, and know where to find existing backups. Advanced features, such as mirroring data on multiple mail accounts and indexing for faster lookups are planned for the future.

4.2 Operation

When MailBackup starts, either for a scheduled backup or manually by the user, it performs the following basic steps:

1. Requests a user-name and password for MailBackup, which is used to decrypt the email account login information.

2. Decides where the backups should be stored, based on quotas and user-specified policies.

3. Creates a `tar` file of modified that is optionally compressed and encrypted. If using SMTP, we split the file to fit into the allowable message size on the email server(s).

4. Stores the backup messages using the access method defined for each email server (currently either SMTP or GmailFS). For SMTP, the subject of the email message contains the date, time, and the sequence number of the message within

the backup. The body contains a list of files that were backed up, along with a checksum for the attachment. For GmailFS, we use the file name for the date and time, and store the rest of the information in a separate file.

To restore a data from a backup, MailBackup is executed in *restore mode*. The user provides a list of files or directories, and can optionally provide a date and time of the last known good version. If no time is specified, the latest backed-up version is retrieved; MailBackup then performs the following steps:

1. Scans the messages in the email accounts for the requested files. If a date and time was specified, MailBackup looks for the first version before that time.
2. Downloads the attachments, computes their checksums, and verifies the integrity of each attachment.
3. Merges files that were split due to message size quotas.
4. Decrypts and uncompresses the files as necessary.

5. PROS, CONS, AND REACTIONS

The obvious advantage of CrawlBackup over other backup techniques is its simplicity. In particular, a basic CrawlBackupCGI implementation requires just *seven* lines and CrawlBackupRestore uses just *six* lines of shell script code. Simple implementations usually have fewer errors and are easier to configure because they can be completely understood by system administrators. Another advantage of CrawlBackup is that it is almost impossible for a hacker to gain access to all copies of the data. Even if the original host with the data is compromised, the hacker has no control over the search engines' caches (if an attacker can control all the major search engines it would be an unprecedented disaster). Some search engines (e.g., Google) allow manual removal of URLs from the caches if the hacker can control the pages on the server. Fortunately, many search engines (e.g., Yahoo! and MSN Search) do not allow removal of information from their caches before the next crawl time. This usually leaves enough time to detect the system compromise and recover the data.

A drawback of CrawlBackup is its inability to completely control the backup frequency and the duration of the backup copy storage in the caches. However, this problem is mitigated by (1) automatic backup frequency adjustments by the search engines, (2) no correlation of crawl times between search engines, and (3) different storage times of different search engines. In particular, search engines automatically adjust their crawling time interval for Web sites that change frequently. There are many search engines and therefore the average time between crawling is much smaller than the crawling intervals of individual search engines. There is a wide diversity between search engines policies. Some crawl frequently but keep the data for a short period of time. Others, like `archive.org`, crawl the Internet infrequently but keep the data forever.

Perhaps the most significant drawback of CrawlBackup is that search engines are not obligated to retain cached data. If a search engine decides to purge its cache for any reason, it will not be held responsible for any data loss. However, as we have just noted, there are several search engines, and so it is improbable that the backed up data will be purged from all caches.

MailBackup gives the user more control over backup frequencies and allows versioning, but the backup process is more complex. However, it is easier to compromise a Web-based email account than to gain access to multiple search engine caches, so the data is less secure. Another drawback is that users need to manage their email accounts and be mindful of their respective quotas. This means that they will either need to delete old versions or create new accounts to increase space.

Creating multiple accounts raises the question of how the companies who provide these free Web services will react to mail accounts and search engine caches being used for data backups. We speculate that search companies would not want their caches being used to store user data, and may try to avoid caching it in the future. However, we believe that creating a small number of email accounts for storing data would not be problematic. In fact, Google is working on GDrive, which would allow users to do just that [9].

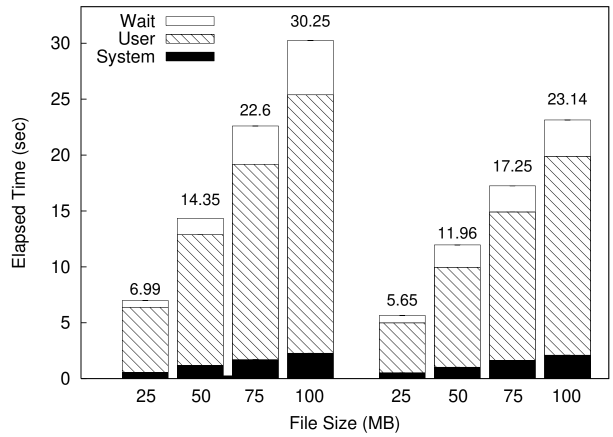

Figure 2: Times for creating backups for various file sizes (left) and restoring the files (right).

6. PERFORMANCE ANALYSIS

We evaluated the overhead of our backup solutions using a hyperthreaded 3.0 GHz Pentium 4 machine with 2GB of RAM running Linux 2.6.14. All experiments used a 7200RPM Serial ATA drive. We ran each test at least ten times and used the Student-t distribution to compute the 95% confidence intervals for the mean elapsed, system, user, and wait times. In each case, the half-widths of the confidence intervals were less than 5% of the mean. Wait time is the elapsed time less CPU time and consists mostly of I/O.

Because CrawlBackup and MailBackup essentially perform the same operations, we created one benchmark for both. We measured how long it took to tar a file, encrypt it and represent the data as ASCII, and then split the file into 10MB chunks. Whereas CrawlBackup explicitly creates an ASCII representation, MailBackup does it when sending the files as email attachments. We chose 10MB because, as Table 1 shows, most popular Webmail clients allow a maximum attachment size of 10MB. We used the Gnu Privacy Guard (GPG) [10] with default settings for encryption and ASCII conversion, which includes use of the CAST-5 encryption algorithm.

The results are shown in the left-hand side of Figure 2. We can see that, as expected, the operations are computationally expensive, and the results are proportional to the file size. For MailBackup, the results represent the amount of time users must wait to backup their data, and for CrawlBackup, the amount of time for the Web server to service the request. It must be noted that encryption is a fairly CPU-intensive operation, and a Web server can be put under heavy load if too many requests arrive at once. Therefore, it is recommended that system administrators restrict access to the CGI scripts to the search engine crawlers and to trusted domains

	CrawlBackup	MailBackup
Simplicity	++	+
Portability	++	+
Flexibility	+	++
Hacker/Virus Resistance	++	+

Table 2: Comparison of the CrawlBackup and MailBackup techniques.

to avoid Denial-of-Service attacks. This is a fairly trivial task with most Web servers. In addition, limiting the amount of simultaneous CrawlBackup script requests would ensure that several bots would not overload the server.

In the right-hand side of Figure 2, we see the times to restore files from a backup. For this benchmark, we reversed the process of the previous benchmark. We see that it takes less time to restore the files for all cases. Fast restore time is important even though backup operations are much more common.

7. PRIOR WORK

Search engine caches have been used to recover previous versions of websites. Warrick attempts to do this automatically [12]. Others have created Windows tools that use free Web-based email services to back up data [11, 17]. These tools simply email specified files to an email address, and do not support recovery, multiple accounts, or backup of only modified files. They also do not split up files to support message size quotas. In addition, some email services do not allow attachments with certain extensions (such as `exe` and `chm`), so it is not always possible to email files. Since MailBackup creates archives, it does not have this problem.

8. CONCLUSIONS

Free Web storage can be effectively used as a backup solution for home users and small businesses. We have investigated two such backup methods: CrawlBackup and MailBackup. Their differences are summarized in Table 2. CrawlBackup, which uses search engine caches to store data, is simpler and more portable than methods that utilize email storage. This is because most email access methods cannot use standard system utilities to manage data. On the other hand, email-based methods provide the user with more options and flexibility. Finally, while a malicious user may be able to delete backups stored on mail servers, search engine caches are less prone to such attacks. Each method has its advantages and disadvantages, but either can be used as an effective, simple, and inexpensive backup solution for a large class of users.

9. ACKNOWLEDGMENTS

We would like to thank the anonymous program committee reviewers for their valuable comments; Margo Seltzer (Harvard) for the accidental motivation of the project; Maxim Lifantsev (Google) for early discussions about the work; and Shachar Shemesh (Lingnu Open Source Consulting Ltd.) for the idea of representing encrypted data as meaningful words.

10. REFERENCES

[1] Backup Direct. Value of business data. `www.backupdirect.net/about_situation_today.htm`, 2004.

[2] J. M. Bennett, M. A. Bauer, and D. Kinchlea. Characteristics of files in NFS environments. *ACM SIGSMALL/PC Notes*, 18(3-4):18–25, 1992.

[3] A. Chervenak, V. Vellanki, and Z. Kurmas. Protecting File Systems: A survey of backup techniques. In *Proc. of Joint IEEE and NASA Mass Storage Conf.*, March 1998.

[4] R. Dad. How the Google Cache can Save Your A$$. *Smart Money Daily*, December 2005. `www.smartmoneydaily.com/Business/How-the-Google-Cache-can-Save-You.aspx`.

[5] N. Haller. The S/KEY One-Time password system. Tech. Rep. RFC 1760, Network Working Group, Feb 1995.

[6] R. Hasan, S. Myagmar, A. Lee, and W. Yurcik. Toward a Threat Model for Storage Systems. In *Proc. of the First ACM Workshop on Storage Security and Survivability (StorageSS 2005)*, pp. 94–102, FairFax, VA, November 2005. ACM.

[7] R. Hasan, W. Yurcik, and S. Myagmar. The Evolution of Storage Service Providers: Techniques and Challenges to Outsourcing Storage. In *Proc. of the First ACM Workshop on Storage Security and Survivability (StorageSS 2005)*, pp. 1–8, FairFax, VA, November 2005

[8] R. Jones. Gmail Filesystem. `http://richard.jones.name/google-hacks/gmail-filesystem/gmail-filesystem.html`.

[9] M. Kane. Going for a GDrive with google. *CNET News*, 7, March 2006. `http://cnet.com/2061-11199_3-6046686.html`.

[10] W. Koch. The GNU privacy guard. `gnupg.org`, Aug 2003.

[11] N. Leghari. GmailSync - Free automated backup solution using Gmail. `http://weblogs.asp.net/nleghari/articles/gmailbackup.aspx`.

[12] F. McCown. Warrick - Tool for Reconstructing a Website. `www.cs.odu.edu/~fmccown/research/lazy/warrick.html`.

[13] E. Riedel, M. Kallahalla, and R. Swaminathan. A Framework for Evaluating Storage System Security. In *Proc. of the First USENIX Conf. on File and Storage Technologies*, pp. 15–30, Monterey, CA, January 2002.

[14] D. Roselli, J. R. Lorch, and T. E. Anderson. A Comparison of File System Workloads. In *Proc. of the Annual USENIX Technical Conf.*, pp. 41–54, San Diego, CA, June 2000.

[15] D. S. Santry, M. J. Feeley, N. C. Hutchinson, A. C. Veitch, R. W. Carton, and J. Ofir. Deciding When to Forget in the Elephant File System. In *Proc. of the 17th ACM Symposium on Operating Systems Principles*, pp. 110–123, Charleston, SC, December 1999

[16] Andrew S. Tanenbaum, Jorrit N. Herder, and Herbert Bos. File size distribution on UNIX systems: then and now. *SIGOPS Operating Systems Review*, 40(1):100–104, 2006.

[17] G. Trapani. Automatically email yourself file backups. `www.lifehacker.com/software/email/geek-to-live-automatically-email-yourself-file-backups-168156.php`.

[18] B. Viksoe. GMail Drive shell extension. `www.viksoe.dk/code/gmail.htm`.

[19] P. Wayner. *Disappearing Cryptography: Being and Nothingness on the Net*. Morgan Kaufmann, April 1996.

[20] E. Zadok, J. Osborn, A. Shater, C. P. Wright, K. Muniswamy-Reddy, and J. Nieh. Reducing Storage Management Costs via Informed User-Based Policies. In *Proc. of the 12th NASA Goddard, 21st IEEE Conf. on Mass Storage Systems and Technologies*, pp. 193–197, College Park, MD, April 2004