Dark Clouds on the Horizon: Using Cloud Storage as Attack Vector and Online Slack Space

Conference Paper · January 2011

CITATIONS
183

READS
653

5 authors, including:

Sebastian Schrittwieser
University of Vienna
82 PUBLICATIONS **1,184** CITATIONS
SEE PROFILE

Manuel Leithner
SBA Research
32 PUBLICATIONS **547** CITATIONS
SEE PROFILE

Edgar Weippl
University of Vienna
332 PUBLICATIONS **4,367** CITATIONS
SEE PROFILE

Dark Clouds on the Horizon: Using Cloud Storage as Attack Vector and Online Slack Space

Martin Mulazzani
SBA Research

Sebastian Schrittwieser
SBA Research

Manuel Leithner
SBA Research

Markus Huber
SBA Research

Edgar Weippl
SBA Research

Abstract

During the past few years, a vast number of online file storage services have been introduced. While several of these services provide basic functionality such as uploading and retrieving files by a specific user, more advanced services offer features such as shared folders, real-time collaboration, minimization of data transfers or unlimited storage space. Within this paper we give an overview of existing file storage services and examine Dropbox, an advanced file storage solution, in depth. We analyze the Dropbox client software as well as its transmission protocol, show weaknesses and outline possible attack vectors against users. Based on our results we show that Dropbox is used to store copyright-protected files from a popular filesharing network. Furthermore Dropbox can be exploited to hide files in the cloud with unlimited storage capacity. We define this as *online slack space*. We conclude by discussing security improvements for modern online storage services in general, and Dropbox in particular. To prevent our attacks cloud storage operators should employ data possession proofs on clients, a technique which has been recently discussed only in the context of assessing trust in cloud storage operators.

1 Introduction

Hosting files on the Internet to make them retrievable from all over the world was one of the goals when the Internet was designed. Many new services have been introduced in recent years to host various type of files on centralized servers or distributed on client machines. Most of today's online storage services follow a very simple design and offer very basic features to their users. From the technical point of view, most of these services are based on existing protocols such as the well known FTP [28], proprietary protocols or WebDAV [22], an extension to the HTTP protocol.

With the advent of cloud computing and the shared usage of resources, these centralized storage services have gained momentum in their usage, and the number of users has increased heavily. In the special case of online cloud storage the shared resource can be disc space on the provider's side, as well as network bandwidth on both the client's and the provider's side. An online storage operator can safely assume that, besides private files as well as encrypted files that are specific and different for every user, a lot of files such as setup files or common media data are stored and used by more than one user. The operator can thus avoid storing multiple physical copies of the same file (apart from redundancy and backups, of course). To the best of our knowledge, Dropbox is the biggest online storage service so far that implements such methods for avoiding unnecessary traffic and storage, with millions of users and billions of files [24]. From a security perspective, however, the shared usage of the user's data raises new challenges. The clear separation of user data cannot be maintained to the same extent as with classic file hosting, and other methods have to be implemented to ensure that within the pool of shared data only authorized access is possible. We consider this to be the most important challenge for efficient and secure "cloud-based" storage services. However, not much work has been previously done in this area to prevent unauthorized data access or information leakage.

We focus our work on Dropbox because it is the biggest cloud storage provider that implements shared file storage on a large scale. New services will offer similar features with cost and time savings on both the client and the operators side, which means that our findings are of importance for all upcoming cloud storage services as well. Our proposed measurements to prevent unauthorized data access and information leakage, exemplarily demonstrated with Dropbox, are not specific to Dropbox and should be used for other online storage services as well. We believe that the number of cloud-based storage

operators will increase heavily in the near future.

Our contribution in this paper is to:

- Document the functionality of an advanced cloud storage service with server-side data deduplication such as Dropbox.
- Show under what circumstances unauthorized access to files stored within Dropbox is possible.
- Assess if Dropbox is used to store copyright-protected material.
- Define online slack space and the unique problems it creates for the process of a forensic examination.
- Explain countermeasures, both on the client and the server side, to mitigate the resulting risks from our attacks for user data.

The remainder of this paper is organized as follows. Related work and the technical details of Dropbox are presented in Section 2. In Section 3 we introduce an attack on files stored at Dropbox, leading to information leakage and unauthorized file access. Section 4 discusses how Dropbox can be exploited by an adversary in various other ways while Section 5 evaluates the feasibility of these attacks. We conclude by proposing various techniques to reduce the attack surface for online storage providers in Section 6.

2 Background

This section describes the technical details and implemented security controls of Dropbox, a popular cloud storage service. Most of the functionality is attributed to the new cloud-paradigm, and not specific to Dropbox. In this paper we use the notion of cloud computing as defined in [9], meaning applications that are accessed over the Internet with the hardware running in a data center not necessarily under the control of the user:

> "Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services." ... "The datacenter hardware and software is what we will call a Cloud."

In the following we describe Dropbox and related literature on cloud storage.

2.1 Dropbox

Since its initial release in September 2008 Dropbox has become one of the most popular cloud storage provider on the Internet. It has 10 million users and stores more then 100 billion files as of May 2011 [2] and saves 1 million files every 5 minutes [3]. Dropbox is mainly an online storage service that can be used to create online backups of files, and one has access to files from any computer or similar device that is connected to the Internet. A desktop client software available for different operating systems keeps all the data in a specified directory in sync with the servers, and synchronizes changes automatically among different client computers by the same user. Subfolders can be shared with other Dropbox users, and changes in shared folders are synced and pushed to every Dropbox account that has been given access to that shared folder. Large parts of the Dropbox client are written in Python.

Internally, Dropbox does not use the concept of files, but every file is split up into chunks of up to 4 megabytes in size. When a user adds a file to his local Dropbox folder, the Dropbox client application calculates the hash values of all the chunks of the file using the SHA-256 algorithm [19]. The hash values are then sent to the server and compared to the hashes already stored on the Dropbox servers. If a file does not exist in their database, the client is requested to upload the chunks. Otherwise the corresponding chunk is not sent to the server because a copy is already stored. The existing file on the server is instead linked to the Dropbox account. This approach allows Dropbox to save traffic and storage costs, and users benefit from a faster syncing process if files are already stored on the Dropbox servers. The software uses numerous techniques to further enhance efficiency e.g., delta encoding, to only transfer those parts of the files that have been modified since the last synchronization with the server. If by any chance two distinct files should have the same hash value, the user would be able to access other users content since the file stored on the servers is simply linked to the users Dropbox account. However, the probability of a coincidental collision in SHA-256 is negligibly small.

The connections between the clients and the Dropbox servers are secured with SSL. Uploaded data is encrypted with AES-256 and stored on Amazons S3 storage service that is part of the Amazon Web Services (AWS) [1]. The AES key is user independent and only secures the data during storage at Amazon S3, while transfer security relies on SSL. Our research on the transmission protocol showed that data is directly sent to Amazon EC2 servers. Therefore, encryption has to be done by EC2 services. We do not know where the keys are stored and if different keys are used for each file chunk. However, the fact that encryption and storage is done at the same place seems questionable to us, as

Amazon is most likely able to access decryption keys [1].

After uploading the chunks that were not yet in the Dropbox storage system, Dropbox calculates the hash values on their servers to validate the correct transmission of the file, and compares the values with the hash values sent by the client. If the hash values do not match, the upload process of the corresponding chunk is repeated. The drawback of this approach is that the server can only calculate the hash values of actually uploaded chunks; it is not able to validate the hash values of files that were already on Dropbox and that were provided by the client. Instead, it *trusts the client software* and links the chunk on the server to the Dropbox account. Therefore, spoofing the hash value of a chunk added to the local Dropbox folder allows a malicious user to access files of other Dropbox users, given that the SHA-256 hash values of the file's chunks are known to the attacker.

Due to the recent buzz in cloud computing many companies compete in the area of cloud storage. Major operating system companies have introduced their services with integration into their system, while small startups can compete by offering cross-OS functionality or more advanced security features. Table 1 compares a selection of popular file storage providers without any claim for completeness. Note that "encrypted storage" means that the file is encrypted locally before it is sent to the cloud storage provider and shared storage means that it is possible to share files and folders between users.

2.2 Related Work

Related work on secure cloud storage focuses mainly on determining if the cloud storage operator is still in possession of the client's file, and if it has been modified. An interesting survey on the security issues of cloud computing in general can be found in [30]. A summary of attacks and new security problems that arise with the usage of cloud computing has been discussed in [17]. In a paper by Shacham et al. [11] it was demonstrated that it is rather easy to map the internal infrastructure of a cloud storage operator. Furthermore they introduced co-location attacks where they have been able to place a virtual machine under their control on the same hardware as a target system, resulting in information leakage and possible side-channel attacks on a virtual machine.

[1] Independently found and confirmed by Christopher Soghoian [5] and Ben Adida [4]

Early publications on file retrievability [25, 14] check if a file can be retrieved from an untrusted third party without retransmitting the whole file. Various papers propose more advanced protocols [11, 12, 20] to ensure that an untrusted server has the original file without retrieving the entire file, while maintaining an overall overhead of $O(1)$. Extensions have been published that allow checking of dynamic data, for example Wang et al. [32] use a Merkle hash tree which allows a third party auditor to audit for malicious providers while allowing public verifiability as well as dynamic data operations. The use of algebraic signatures was proposed in [29], while a similar approach based on homomorphic tokens has been proposed in [31]. Another cryptographic tree structure is named "Cryptree" [23] and is part of the Wuala online storage system. It allows strong authentication by using encryption and can be used for P2P networks as well as untrusted cloud storage. The HAIL system proposed in [13] can be seen as an implementation of a service-oriented version of RAID across multiple cloud storage operators.

Harnik et al. describe similar attacks in a recent paper [24] on cloud storage services which use server-side data deduplication. They recommend using encryption to stop server-side data deduplication, and propose a randomized threshold in environments where encryption is undesirable. However, they do not employ client-side data possession proofs to prevent hash manipulation attacks, and have no practical evaluation for their attacks.

3 Unauthorized File Access

In this section we introduce three different attacks on Dropbox that enable access to arbitrary files given that the hash values of the file, respectively the file chunks, are known. If an arbitrary cloud storage service relies on the client for hash calculation in server-side data deduplication implementations, these attacks are applicable as well.

3.1 Hash Value Manipulation Attack

For the calculation of SHA-256 hash values, Dropbox does not use the `hashlib` library which is part of Python. Instead it delegates the calculation to OpenSSL [18] by including a wrapper library called NCrypto [6]. The Dropbox clients for Linux and Mac OS X dynamically link to libraries such as NCrypto and do not verify their integrity before using them. We modified the publicly available source code of NCrypto so that it replaces the hash value that was calculated by OpenSSL with our own value (see Figure 1), built it

Name	Protocol	Encrypted transmission	Encrypted storage	Shared storage
Dropbox	proprietary	yes	no	yes
Box.net	proprietary	yes	yes (enterprise only)	yes
Wuala	Cryptree	yes	yes	yes
TeamDrive	many	yes	yes	yes
SpiderOak	proprietary	yes	yes	yes
Windows Live Skydrive	WebDAV	yes	no	yes
Apple iDisk	WebDAV	no	no	no
Ubuntu One	u1storage	yes	no	yes

Table 1: Online Storage Providers

and replaced the library that was shipped with Dropbox. The Dropbox client does not detect this modification and transmits for any new file in the local Dropbox the modified hash value to the server. If the transmitted hash value does not exist in the server's database, the server requests the file from the client and tries to verify the hash value after the transmission. Because of our manipulation on the client side, the hash values will not match and the server would detect that. The server would then re-request the file to overcome an apparent transmission error.

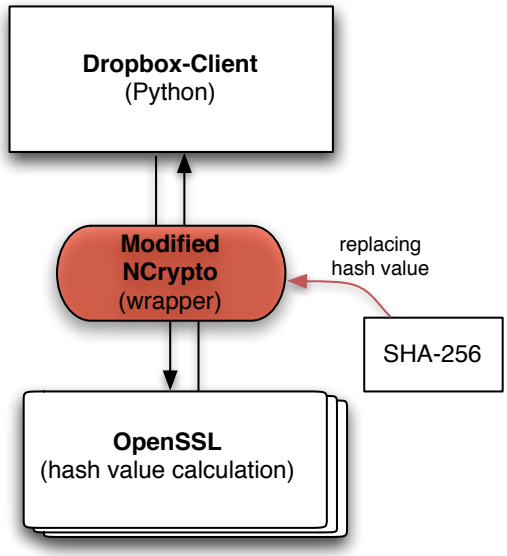

Figure 1: Hash Value Manipulation Attack

However, if the hash value is already in the server's databases the server trusts the hash value calculation of the client and does not request the file from the client. Instead it links the corresponding file/chunk to the Dropbox account. Due to the manipulation of the hash value we thus got unauthorized access to arbitrary files.

This attack is completely undetectable to the user. If the attacker already knows the hash values, he can download files directly from the Dropbox server and no interaction with the client is needed which could be logged or detected on the client side. The victim is unable to notice this in any way, as no access to his computer is required. Even for the Dropbox servers this unauthorized access to arbitrary files is not detectable because they believe the attacker already owns the files, and simply added them to their local Dropbox folder.

3.2 Stolen Host ID Attack

During setup of the Dropbox client application on a computer or smartphone, a unique **host ID** is created which links that specific device to the owner's Dropbox account. The client software does not store username and password. Instead, the host ID is used for client and user authentication. It is a random looking 128-bit key that is calculated by the Dropbox server from several seeding values provided by the client (e.g. username, exact date and time). The algorithm is not publicly known. This linking requires the user's account credentials. When the client on that host is successfully linked, no further authentication is required for that host as long as the Dropbox software is not removed.

If the host ID is stolen by an attacker, extracted by malware or by social engineering, all the files on that users accounts can be downloaded by the attacker. He simply replaces his own host ID with the stolen one, resyncs Dropbox and consequently downloads every file.

3.3 Direct Download Attack

Dropbox's transmission protocol between the client software and the server is built on HTTPS. The client software can request file chunks from https://dl-clientXX.dropbox.com/retrieve (where XX is replaced by consecutive numbers) by submitting the SHA-256 hash value of the file chunk and a valid host ID as HTTPS POST data. Surprisingly, the host ID doesn't even need to be linked to a Dropbox account that owns

the corresponding file. Any valid host ID can be used to request a file chunk as long as the hash value of the chunk is known and the file is stored at Dropbox. As we will see later, Dropbox hardly deletes any data. It is even possible to just create an HTTPS request with any valid host ID, and the hash value of the chunk to be downloaded. This approach could be easily detected by Dropbox because a host ID that was not used to upload a chunk or is known to be in possession of the chunk would try to download it. By contrast the hash manipulation attack described above is undetectable for the Dropbox server, and (minor) changes to the core communication protocol would be needed to detect it.

3.4 Attack Detection

To sum up, when an attacker is able to get access to the content of the client database, he is able to download all the files of the corresponding Dropbox account directly from the Dropbox servers. No further access to the victim's system is needed, and in the simplest case only the host ID needs to be sent to the attacker. An alternative approach for the attacker is to access only specific files, by obtaining only the hash values of the file. The owner of the files is unable to detect that the attacker accessed the files, for all three attacks. From the cloud storage service operators point of view, the stolen host-ID attack as well as the direct download attack are detectable to some extent. We discuss some countermeasures in section 6. However, by using the hash manipulation attack the attacker can avoid detection completely, as this form of unauthorized access looks like the attacker already owns the file to Dropbox. Table 2 gives an overview of all of the different attacks that can lead to unauthorized file access and information leakage [2].

4 Attack Vectors and Online Slack Space

This section discusses known attack techniques to exploit cloud storage and Dropbox on a large scale. It outlines already known attack vectors, and how they could be used with the help of Dropbox, or any other cloud storage service with weak security. Most of them can have a severe impact and should be considered in the threat model of such services.

[2]We communicated with Dropbox and reported our findings prior to publishing this paper. They implemented a temporary fix to prevent these types of attacks and will include a permanent solution in future versions.

4.1 Hidden Channel, Data Leakage

The attacks discussed above can be used in numerous ways to attack clients, for example by using Dropbox as a drop zone for important and possibly sensitive data. If the victim is using Dropbox (or any other cloud storage services which is vulnerable to our discovered attack) these services might be used to exfiltrate data a lot stealthier and faster with a covert channel than using regular covert channels [16]. The amount of data that needs to be sent over the covert channel would be reduced to a single host ID or the hash values of specific files instead of the full file. Furthermore the attacker could copy important files to the Dropbox folder, wait until they are stored on the cloud service and delete them again. Afterwards he transmits the hash values to the attacker and the attacker then downloads these files directly from Dropbox. This attack requires that the attacker is able to execute code and has access to the victim's file system e.g. by using malware. One might argue that these are tough preconditions for this scenario to work. However, as in example, in the case of corporate firewalls this kind of data leakage is much harder to detect as all traffic with Dropbox is encrypted with SSL and the transfers would blend in perfectly with regular Dropbox activity, since Dropbox itself is used for transmitting the data. Currently the client has no control measures to decide upon which data might get stored in the Dropbox folder. The scheme for leaking information and transmitting data to an attacker is depicted in Figure 2.

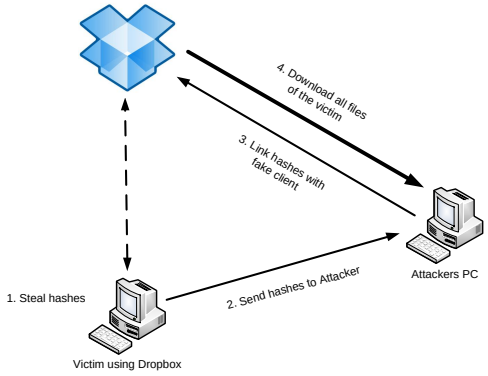

Figure 2: Covert Channel with Dropbox

4.2 Online Slack Space

Uploading a file works very similarly to downloading with HTTPS (as described above, see section 3.3). The client software uploads a chunk to Dropbox by calling https://dl-clientXX.dropbox.com/store with the hash value and the host ID as HTTPS POST data along with the actual data. After the upload is finished, the client

Method	Detectability	Consequences
Hash Value Manipulation Attack	Undetectable	Unauthorized file access
Direct Download Attack	Dropbox only	Unauthorized file access
Stolen Host ID Attack	Dropbox only	Get all user files

Table 2: Variants of the Attack

software links the uploaded files to the host ID with another HTTPS request. The updated or newly added files are now pushed to all computers of the user, and to all other user accounts if the folder is a shared folder.

A modified client software can upload files without limitation, if the linking step is omitted. Dropbox can thus be used to store data without decreasing the available amount of data. We define this as *online slack space* as it is similar to regular slack space [21] from the perspective of a forensic examiner where information is hidden in the last block of files on the filesystem that are not using the entire block. Instead of hiding information in the last block of a file, data is hidden in Dropbox chunks that are not linked to the attackers account. If used in combination with a live CD operating system, no traces are left on the computer that could be used in the forensic process to infer the existence of that data once the computer is powered down. We believe that there is no limitation on how much information could be hidden, as the exploited mechanisms are the same as those which are used by the Dropbox application.

4.3 Attack Vector

If the host ID is known to an attacker, he can upload and link arbitrary files to the victim's Dropbox account. Instead of linking the file to his account with the second HTTPS request, he can use an arbitrary host ID with which to link the file. In combination with an exploit of the operating system file preview functions, e.g. on one of the recent vulnerabilities in Windows [3], Linux [4], or MacOS [5], this becomes a powerful exploitation technique. An attacker could use any 0-day weakness in the file preview of supported operating systems to execute code on the victim's computer, by pushing a manipulated file into his Dropbox folder and waiting for the user to open that directory. Social engineering could additionally be used to trick the victim into executing a file with a promising filename.

To get access to the host ID in the first place is tricky, and in any case access to the filesystem is needed in the first place. This however does not reduce the consequences, as it is possible to store files remotely in other peoples Dropbox. A large scale infection using Dropbox is however very unlikely, and if an attacker is able to retrieve the host ID he already owns the system.

[3] Windows Explorer: CVE-2010-2568 or CVE-2010-3970

[4] Evince in Nautilus: CVE-2010-2640

[5] Finder: CVE-2006-2277

5 Evaluation

This section studies some of the attacks introduced. We evaluate whether Dropbox is used to store popular files from the filesharing network thepiratebay.org [6] as well as how long data is stored in the previously defined online slack space.

5.1 Stored files on Dropbox

With the hash manipulation attack and the direct download attack described above it becomes possible to test if a given file is already stored on Dropbox. We used that to evaluate if Dropbox is used for storing filesharing files, as filesharing protocols like BitTorrent rely heavily on hashing for file identification. We downloaded the top 100 torrents from thepiratebay.org [7] as of the middle of September 2010. Unfortunately, BitTorrent uses SHA-1 hashes to identify files and their chunks, so the information in the .torrent file itself is not sufficient and we had to download parts of the content. As most of the files on BitTorrent are protected by copyright, we decided to download every file from the .torrent that lacks copyright protection to protect us from legal complaints, but are still sufficient to prove that Dropbox is used to store these kind of files. To further proctect us against complaints based on our IP address, our BitTorrent client was modified to prevent upload of any data, as described similarly in [27]. We downloaded only the first 4 megabytes of any file that exceeds this size, as the first chunk is already sufficient to tell if a given file is stored on Dropbox or not using the hash manipulation attack.

We observed the following different types of files that were identified by the .torrent files:

- Copyright protected content such as movies, songs or episodes of popular series.
- "Identifying files" that are specific to the copyright protected material, such as sample files, screen captures or checksum files, but without copyright.

[6] Online at `http://thepiratebay.org`

- Static files that are part of many torrents, such as release group information files or links to websites.

Those "identifying files" we observed had the following extensions and information:

- *.nfo*: Contains information from the release group that created the .torrent e.g., list of files, installation instructions or detailed information and ratings for movies.
- *.srt*: Contains subtitles for video files.
- *.sfv*: Contains CRC32 checksums for every file within the .torrent.
- *.jpg*: Contains screenshots of movies or album covers.
- *.torrent*: The torrent itself contains the hash values of all the files, chunks as well as necessary tracker information for the clients.

In total from those top 100 torrent archives, 98 contained identifying files. We removed the two .torrents from our test set that did not contain such identifying files. 24 hours later we downloaded the newest entries from the top 100 list, to check how long it takes from the publication of a torrent until it is stored on Dropbox. 9 new torrents, mostly series, were added to the test set. In Table 3 we show in which categories they where categorized by thepiratebay.org.

Category	Quantity
Application	3
Game	5
Movie	64
Music	6
Series	29
Sum	107

Table 3: Distribution of tested .torrents

When we downloaded the "identifying files" from these 107 .torrent, they had in total approximately 460k seeders and 360k leechers connected (not necessarily disjoint), with the total number of complete downloads possibly much higher. For every .torrent file and every identifying file from the .torrent's content we generated the sha256 hash value and checked if the files were stored on Dropbox, in total 368 hashes. If the file was bigger then 4 megabytes, we only generated the hash of the first chunk. Our script did not use the completely stealthy approach described above, but the less stealthy approach by creating an HTTPS request with a valid host ID as the overall stealthiness was in our case not an issue.

From those 368 hashes, 356 files were retrievable, only 12 hashes were unknown to Dropbox and the corresponding files were not stored on Dropbox. Those 12 files were linked to 8 .torrent files. The details:

- In one case the identifying file of the .torrent was not on Dropbox, but the .torrent file was.
- In three cases the .torrent file was not on Dropbox, but the identifying files were.
- In four cases the .nfo file was not on Dropbox, but other iIn fact, it might be the case that only one person uses Dropbox to store these files. dentifying files from the same .torrent were.

This means that for every .torrent either the .torrent file, the content or both are easily retrievable from Dropbox once the hashes are known. Table 4 shows the numbers in details, where hit rate describes how many of them were retrievable from Dropbox.

File	Quantity	Hitrate	Hitrate rel.
.torrent:	107	106	99%
.nfo:	53	49	92%
others:	208	201	97%
In total:	368	356	97%

Table 4: Hit rate for filesharing

Furthermore we analyzed the age of the .torrents to see how quick Dropbox users are to download the .torrents and the corresponding content, and to upload everything to Dropbox. Most of the .torrent files were relatively young, as approximately 20 % of the top 100 .torrent files were less than 24 hours on piratebay before we were able to retrieve them from Dropbox. Figure 3 shows the distribution of age from all the .torrents:

5.2 Online Slack Space Evaluation

To assess if Dropbox could be used to hide files by uploading without linking them to any user account, we generated a set of 30 files with random data and uploaded them with the HTTPS request method. Furthermore we uploaded 55 files with a regular Dropbox account and deleted them right afterwards, to assess if Dropbox ever deletes old user data. We furthermore evaluated if there is some kind of garbage collection that removes files after a given threshold of time since the upload. The files were then downloaded every 24 hours and checked for consistency by calculating multiple hash functions and comparing the hashvalues. By using multiple files with various sizes and random content we minimized the likelihood of an unintended hash collision and avoided testing for a file that is stored by another user and thus

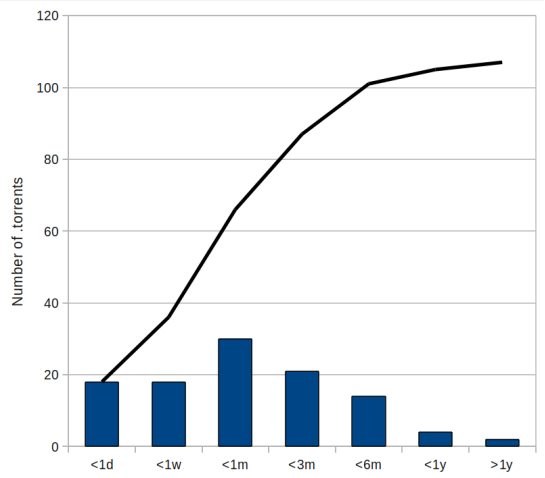

Figure 3: Age of .torrents

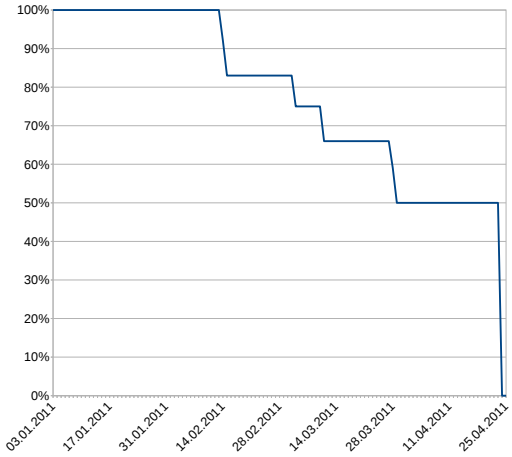

Figure 4: Online slack without linking over time

always retrievable. Table 5 summarizes the setup.

Method of upload	#	Testduration	Hitrate
Regular folder	25	6 months	100%
Shared folder	30	6 months	100%
HTTPS request	30	>3 months	50%
In total:	85	—	**100%**

Table 5: Online slack experiments

Long term undelete: With the free account users can undo file modifications or undelete files through the webinterface from the last 30 days. With a so called "Pro" account (where the users pay for additional storage space and other features) undelete is available for all files and all times. We uploaded 55 files in total on October 7th 2010, 30 files in a shared folder with another Dropbox account and 25 files in an unshared folder. Until Dropbox fixed the HTTPS download attack at the end of April 2011, 100% have been constantly available. More then 6 months after uploading, all files were still retrievable, without exception.

Online slack: We uploaded 30 files of various sizes without linking them to any account with the HTTPS method at the beginning of January 2011. More then 4 weeks later, all files were still retrievable. When Dropbox fixed the HTTPS download attack in late April 2011, 50% of the files were still available. See Figure 4 for details.

5.3 Discussion

It surprised us that from every .torrent file, either the .torrent, the content or both could be retrieved from Dropbox, especially considering that some of the .torrent files were only a few hours created before we retrieved them. 97% means that Dropbox is heavily used for storing files from filesharing networks. It is also interesting to note that some of the .torrent files contained more content regarding storage space than the free Dropbox account currently offers (2 gigabytes at the time of writing). 11 out of the set of tested 107 .torrents contained more then 2 gigabytes as they were DVD images, the biggest with 7.2 gigabytes in total size. This means that whoever stored those files on Dropbox has either a Dropbox Pro account (for which he or she pays a monthly fee), or that he invited a lot of friends to get additional storage space from the Dropbox referral program.

However, we could only infer the existence of these files. With the approach we used it is not possible to quantify to what extent Dropbox is used for filesharing among multiple users. Our results only show that within the last three to six months at least one Bittorrent user saved his downloads in Dropbox, respectively that since the .torrent has been created. No conclusions can be drawn as to whether they are saved in shared folders, or if only one person or possibly thousands of people uses Dropbox in that way. In fact, it is equally likely that a single person uses Dropbox to store these files.

With our experiments regarding online slack space we showed that it is very easy to hide data on Dropbox with low accountability. It becomes rather trivial to get some of the advanced features of Dropbox like unlimited undelete and versioning, without costs. Furthermore a malicious user can upload files without linking them to his account, resulting in possibly unlimited storage space

while at the same time possibly causing problems in a standard forensic examination. In an advanced setup, the examinator might be confronted with a computer that has no harddrive, booting from read only media such as a Linux live CD and saving all files in online slack space. No traces or local evidence would be extractable from the computer [15], which will be an issue in future forensic examinations. This is similar to using the private mode in modern browsers which do not save information locally [8].

6 Keeping the cloud white

To ensure trust in cloud storage operators it is vital to not only make sure that the untrusted cloud storage operator keeps the files secure with regards to availability [25], but also to ensure that the client cannot get attacked with these services. We provide generic security recommendations for all storage providers to prevent our attacks, and propose changes to the communication protocol of Dropbox to include data possession proofs that can be precalculated on the cloud storage operato'rs side and implemented efficiently as database lookups.

6.1 Basic security primitives

Our attacks are not only applicable to Dropbox, but to all cloud storage services where a server-side data deduplication scheme is used to prevent retransmission of files that are already stored at the provider. Current implementations are based on simple hashing. However, the client software cannot be trusted to calculate the hash value correctly and a stronger proof of ownership is needed. This is a new security aspect of cloud computing, as up till now mostly trust in the service operator was an issue, and not the client.

To ensure that the client is in possession of a file, a strong protocol for provable data possession is needed, based on either cryptography or probabilistic proofs or both. This can be done by using a recent provable data possession algorithm such as [11], where the cloud storage operator selects which challenges the client has to answer to get access to the file on the server and thus omit the retransmission which is costly for both the client and the operator. Recent publications proposed different approaches with varying storage and computational overhead [12, 20, 10]. Furthermore every service should use SSL for all communication and data transfers, something which we observed was not the case with every service.

6.2 Secure Dropbox

To fix the discovered security issues in Dropbox we propose several steps to mitigate the risk of abuse. First of all, a secure **data possession protocol** should be used to prevent the clients to get access to files only by knowing the hash value of a file. Eventually every cloud storage operator should employ such a protocol if the client is not part of a trusted environment. We therefore propose the implementation of a simple challenge-response mechanism as outlined in Fig. 5. In essence: If the client transmits a hash value already known to the storage operator, the server has to verify if the client is in possession of the entire file or only the hash value. The server could do so by requesting randomly chosen bytes from the data during the upload process. Let H be a cryptographic hash function which maps data D of arbitrary length to fixed length hash value.
$Push_{init}(U, p(U), H(D))$ is a function that initiates the upload of data D from the client to the server. The user U and an authentication token $p(U)$ are sent along with the hash value $H(D)$ of data D. $Push(U, p(U), D)$ is the actual uploading process of data D to the server. $Req(U, p(U), H(D))$ is a function that requests data D from the server.
$Ver(Ver_{off}, H(D))$ is a function that requests randomly chosen bytes from data D by specifying their offsets in the array Ver_{off}.

Uploading **chunks without linking** them to a users

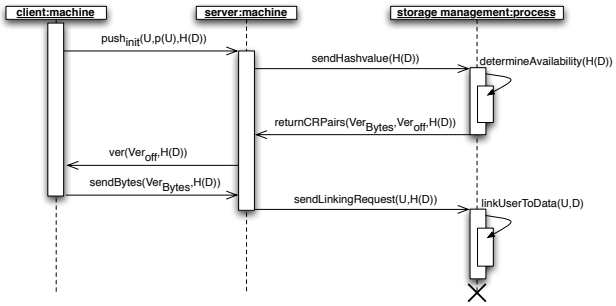

Figure 5: Data verification during upload

Dropbox should not be allowed, on the one hand to prevent clients to have unlimited storage capacity, on the other hand to make online slack space on Dropbox infeasible. In many scenarios it is still cheaper to just add storage capacity instead of finding a reliable metric on what data to delete - however, to prevent misuse of historic data and online slackspace, all chunks that are not linked to a file that is retrievable by a client should be deleted.

To further enhance security several behavioral aspects

Security Measure	Consequences
1. Data possession protocol	Prevent hash manipulation attacks
2. No chunks without linking	Defy online slack space
3. Check for host ID activity	Prevent access if host is not online
4. Dynamic host ID	Smaller window of opportunity
5. Enforcement of data ownership	No unauthorized data access

Table 6: Security Improvements for Dropbox

can be leveraged, for example to **check for host ID activity** - if a client turns on his computer he connects to Dropbox to see if any file has been updated or new files were added. Afterwards, only that IP address should be allowed to download files from that host IDs Dropbox. If the user changes IP e.g., by using a VPN or changing location, Dropbox needs to rebuild the connection anyway and could use that to link that host ID to that specific IP. In fact, the host ID should be used like a cookie [26] if used for authentication, dynamic in nature and changeable. A **dynamic host ID** would reduce the window of opportunity that an attacker could use to clone a victim's Dropbox by stealing the host ID. Most importantly, Dropbox should keep track of which files are in which Dropboxes (**enforcement of data ownership**). If a client downloads a chunk that has not been in his or her Dropbox, this is easily detectable for Dropbox.

Unfortunately we are unable to assess the performance impact and communication overhead of our mitigation strategies, but we believe that most of them can be implemented as simple database lookups. Different data possession algorithms have already been studied for their overhead, for example S-PDP and E-PDP from [11] are bounded by $O(1)$. Table 6 summarizes all needed mitigation steps to prevent our attacks.

7 Conclusion

In this paper we presented specific attacks on cloud storage operators where the attacker can download arbitrary files under certain conditions. We proved the feasibility on the online storage provider Dropbox and showed that Dropbox is used heavily to store data from thepiratebay.org, a popular BitTorrent website. Furthermore we defined and evaluated online slack space and demonstrated that it can be used to hide files. We believe that these vulnerabilities are not specific to Dropbox, as the underlying communication protocol is straightforward and very likely to be adopted by other cloud storage operators to save bandwidth and storage overhead. The discussed countermeasures, especially the data possession proof on the client side, should be included by all cloud storage operators.

Acknowledgements

We would like to thank Arash Ferdowsi and Lorcan Morgan for their helpful comments. Furthermore we would like to thank the reviewers for their feedback. This work has been supported by the Austrian Research Promotion Agency under grant 825747 and 820854.

References

[1] Amazon.com, Amazon Web Services (AWS). Online at http://aws.amazon.com.

[2] At Dropbox, Over 100 Billion Files Served–And Counting, retrieved May 23rd, 2011. Online at http://gigaom.com/2011/05/23/at-dropbox-over-100-billion-files-served-and-counting/.

[3] Dropbox Users Save 1 Million Files Every 5 Minutes, retrieved May 24rd, 2011. Online at http://mashable.com/2011/05/23/dropbox-stats/.

[4] Grab the pitchforks!... again, retrieved April 19th, 2011. Online at http://benlog.com/articles/2011/04/19/grab-the-pitchforks-again/.

[5] How Dropbox sacrifices user privacy for cost savings, retrieved April 12th, 2011. Online at http://paranoia.dubfire.net/2011/04/how-dropbox-sacrifices-user-privacy-for.html.

[6] NCrypto Homepage, retrieved June 1st, 2011. Online at http://ncrypto.sourceforge.net/.

[7] Piratebay top 100. Online at http://thepiratebay.org/top/all.

[8] AGGARWAL, G., BURSZTEIN, E., JACKSON, C., AND BONEH, D. An analysis of private browsing modes in modern browsers. In *Proceedings of the 19th USENIX conference on Security* (2010), USENIX Security'10.

[9] ARMBRUST, M., FOX, A., GRIFFITH, R., JOSEPH, A. D., KATZ, R., KONWINSKI, A., LEE, G., PATTERSON, D., RABKIN, A., STOICA, I., AND ZAHARIA, M. A view of cloud computing. *Communications of the ACM 53*, 4 (2010), 50–58.

[10] ATENIESE, G., BURNS, R., CURTMOLA, R., HERRING, J., KHAN, O., KISSNER, L., PETERSON, Z., AND SONG, D. Remote data checking using provable data possession. *ACM Transactions on Information and System Security (TISSEC) 14*, 1 (2011), 12.

[11] ATENIESE, G., BURNS, R., CURTMOLA, R., HERRING, J., KISSNER, L., PETERSON, Z., AND SONG, D. Provable data possession at untrusted stores. In *Proceedings of the 14th ACM conference on Computer and communications security* (2007), CCS '07, ACM, pp. 598–609.

[12] ATENIESE, G., DI PIETRO, R., MANCINI, L., AND TSUDIK, G. Scalable and Efficient Provable Data Possession. In *Proceedings of the 4th international conference on Security and privacy in communication netowrks* (2008), ACM, pp. 1–10.

[13] BOWERS, K., JUELS, A., AND OPREA, A. HAIL: A high-availability and integrity layer for cloud storage. In *Proceedings of the 16th ACM conference on Computer and communications security* (2009), ACM, pp. 187–198.

[14] BOWERS, K., JUELS, A., AND OPREA, A. Proofs of retrievability: Theory and implementation. In *Proceedings of the 2009 ACM workshop on Cloud computing security* (2009), ACM, pp. 43–54.

[15] BREZINSKI, D., AND KILLALEA, T. Guidelines for Evidence Collection and Archiving (RFC 3227). *Network Working Group, The Internet Engineering Task Force* (2002).

[16] CABUK, S., BRODLEY, C. E., AND SHIELDS, C. Ip covert timing channels: design and detection. In *Proceedings of the 11th ACM conference on Computer and communications security* (2004), CCS '04, pp. 178–187.

[17] CHOW, R., GOLLE, P., JAKOBSSON, M., SHI, E., STADDON, J., MASUOKA, R., AND MOLINA, J. Controlling data in the cloud: outsourcing computation without outsourcing control. In *Proceedings of the 2009 ACM workshop on Cloud computing security* (2009), ACM, pp. 85–90.

[18] COX, M., ENGELSCHALL, R., HENSON, S., LAURIE, B., YOUNG, E., AND HUDSON, T. Openssl, 2001.

[19] EASTLAKE, D., AND HANSEN, T. US Secure Hash Algorithms (SHA and HMAC-SHA). Tech. rep., RFC 4634, July 2006.

[20] ERWAY, C., KÜPCÜ, A., PAPAMANTHOU, C., AND TAMASSIA, R. Dynamic Provable Data Possession. In *Proceedings of the 16th ACM conference on Computer and communications security* (2009), ACM, pp. 213–222.

[21] GARFINKEL, S., AND SHELAT, A. Remembrance of data passed: A study of disk sanitization practices. *Security & Privacy, IEEE 1*, 1 (2003), 17–27.

[22] GOLAND, Y., WHITEHEAD, E., FAIZI, A., CARTER, S., AND JENSEN, D. HTTP Extensions for Distributed Authoring–WEBDAV. *Microsoft, UC Irvine, Netscape, Novell. Internet Proposed Standard Request for Comments (RFC) 2518* (1999).

[23] GROLIMUND, D., MEISSER, L., SCHMID, S., AND WATTENHOFER, R. Cryptree: A folder tree structure for cryptographic file systems. In *Reliable Distributed Systems, 2006. SRDS'06. 25th IEEE Symposium on* (2006), IEEE, pp. 189–198.

[24] HARNIK, D., PINKAS, B., AND SHULMAN-PELEG, A. Side channels in cloud services: Deduplication in cloud storage. *Security & Privacy, IEEE 8*, 6 (2010), 40–47.

[25] JUELS, A., AND KALISKI JR, B. PORs: Proofs of retrievability for large files. In *Proceedings of the 14th ACM conference on Computer and communications security* (2007), ACM, pp. 584–597.

[26] KRISTOL, D. HTTP Cookies: Standards, privacy, and politics. *ACM Transactions on Internet Technology (TOIT) 1*, 2 (2001), 151–198.

[27] PIATEK, M., KOHNO, T., AND KRISHNAMURTHY, A. Challenges and directions for monitoring P2P file sharing networks-or: why my printer received a DMCA takedown notice. In *Proceedings of the 3rd conference on Hot topics in security* (2008), USENIX Association, p. 12.

[28] POSTEL, J., AND REYNOLDS, J. RFC 959: File transfer protocol. *Network Working Group* (1985).

[29] SCHWARZ, T., AND MILLER, E. Store, forget, and check: Using algebraic signatures to check remotely administered storage. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on* (2006), IEEE, p. 12.

[30] SUBASHINI, S., AND KAVITHA, V. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications* (2010).

[31] WANG, C., WANG, Q., REN, K., AND LOU, W. Ensuring data storage security in cloud computing. In *Quality of Service, 2009. IWQoS. 17th International Workshop on* (2009), Ieee, pp. 1–9.

[32] WANG, Q., WANG, C., LI, J., REN, K., AND LOU, W. Enabling public verifiability and data dynamics for storage security in cloud computing. *Computer Security–ESORICS 2009* (2010), 355–370.