Cloudsweeper: Enabling Data-Centric Document Management for Secure Cloud Archives

Peter Snyder and Chris Kanich
University of Illinois at Chicago
Chicago, Illinois, USA
{psndye2,ckanich}@uic.edu

ABSTRACT

Cloud based storage accounts like web email are compromised on a daily basis. At the same time, billions of Internet users store private information in these accounts. As the Internet matures and these accounts accrue more information, these accounts become a single point of failure for both users' online identities and large amounts of their private information. This paper presents two contributions: the first, the *heterogeneous documents* abstraction, is a data-centric strategy for protecting high value information stored in globally accessible storage. Secondly, we present Cloudsweeper, an implementation of the heterogeneous documents strategy as a cloud-based email protection system. Cloudsweeper gives users the opportunity to remove or "lock up" sensitive, unexpected, and rarely used information to mitigate the risks of cloud storage accounts without sacrificing the benefits of cloud storage or computation. We show that Cloudsweeper can efficiently assist users in pinpointing and protecting passwords emailed to them in cleartext. We present performance measurements showing that the system can rewrite past emails stored at cloud providers quickly, along with initial results regarding user preferences for redacted cloud storage.

Categories and Subject Descriptors

H.3.2 [**Information Systems Applications**]: Information Storage; D.4.6 [**Operating Systems**]: Security and Protection; C.2.0 [**Computer-Communication Networks**]: General

Keywords

Cloud Storage; Data Loss Prevention; Attack Mitigation

1. INTRODUCTION

Modern internet and computing infrastructure has matured to the point that any data recorded today is likely to be retained and accessible for an extended period of time. Inaccessibility due to changing formats and media is less likely as the number and quality of available libraries, operating systems, and tools has grown. Global connectivity allows users to easily store decades of documents in the cloud, at home, or even on a mobile device. Modern, efficient search makes referencing an individual email or document only a query away, and cloud storage effortlessly synchronizes all of one's documents between different devices.

Less clear, however, is the risk that accompanies these monolithic archives. In many cases, the only thing that stands between a cybercriminal and years of documents is a single password: easily phished, purchased, or recovered from a database breach. When someone lives their entire life using cloud-based storage that effortlessly synchronizes between old and new devices, it is likely that extremely sensitive documents are accessible within this archive. While today's cybercriminals typically exploit the capabilities of a stolen account to send spam or defraud friends, the gargantuan volume of data stored within these accounts makes them more lucrative to attackers, and merits a data-centric approach to archive management.

Instant global access to decades of information upends most of the assumptions underlying traditional strategies for securing our personal effects. Older methods like safe deposit boxes or home safes provide reasonable security at the point of attack because they can only be accessed by someone standing next to them. While this physical locality constraint can be seen as a disadvantage with respect to accessibility, the current "access to everything all of the time from anywhere" policy exposed by most online services exposes users to the risk of having their entire archive stolen in the event of an account compromise.

Rather than continuing to focus efforts on security at the perimeter with different password schemes or multi-factor authentication, a complementary approach would be to investigate changes to availability that can minimize risk while maintaining the utility of a globally connected archive. Developing a deeper understanding of the risks of data compromise alongside a set of mechanisms and policies that provide data-centric security will allow users to assess and manage the risks involved in maintaining a full life archive.

Cloudsweeper implements the heterogeneous documents abstraction, by allowing users to automatically protect high-value but low-utility information within their cloud based email account. Heterogeneous documents allow users to place an extra level of protection on a limited portion of a document stored in the cloud. The limited, sensitive portions of the documents are protected by a key, stored separately from the cloud archive, so that compromise of the cloud

CCSW'13, November 8, 2013, Berlin, Germany.

ACM 978-1-4503-2490-8/13/11 ...$15.00.
http://dx.doi.org/10.1145/2517488.2517495.

archive does not give access to protected information. By allowing cloud based computation like search to continue functioning even though part of the message is encrypted, Cloudsweeper maintains the utility of the archive both to the service provider and the user.

As an initial exploration of using the heterogeneous documents abstraction, we built a service to automatically find and redact passwords which have been emailed to users in plaintext. Very often, these passwords are the epitome of high-risk, low utility information: users most likely don't want these passwords saved in their archive, but might still have use for the email message as a whole. By redacting or encrypting only the password, the utility of the archive is unchanged but the risk is greatly decreased. We have deployed a public version of this tool that works on Gmail accounts which has been used nearly 700 times by Internet users to protect over 25,000 messages.

The remainder of this paper is structured as follows: Section 2 presents related work and contextualizes the heterogeneous documents abstraction with respect to alternative approaches to data protection. Section 3 outlines our threat model and design goals, and Section 4 covers our implementation of the search, redact, and encrypt functions of Cloudsweeper. Section 5 evaluates the performance of Cloudsweeper as a web-based service, with respect to individual user experience and scaling to serve several clients simultaneously. Section 6 discusses our findings and avenues for future work, and Section 7 concludes.

2. BACKGROUND

The strategies underlying heterogeneous documents and Cloudsweeper stem from three domains: personal information management, attacker goal based security, and data exfiltration protection mechanisms. By combining techniques from these three areas, we hope to create effective protection for high risk archives.

2.1 Approaches for protecting long-lived data

Personal information management (PIM), a subfield of information retrieval, studies task and role based retrieval of personal information; Jones aggregated the scholarship in this area in [9]. Email has become a prime target for PIM research, as for many users email has taken over many of the tasks traditionally associated with PIM including task management, archiving, and contact management [23].

While the intended practitioners of personal information management are legitimate users, the insights surrounding strategies for retrieving important information may also apply to malicious users with the same goals in mind. Investigating how successful an attacker might be at extracting useful information from a user's account will allow us to evaluate the risks inherent in long term accessible storage.

Czerwinski et al. raise questions regarding the technological, legal, and social implications of ubiquitous, abundant storage [4]. The authors present an introduction to the vast array of measurable and recordable data that ubiquitous sensors and storage has enabled, and build a taxonomy of the goals and challenges engendered by the situation. While communication, behavioral, and sensor data can all be collected and stored locally, deciding what subset or aggregate data should be kept or shared is an open question.

Usability is a fundamental, primary concern when creating user-facing security mechanisms. Usable security and privacy research typically focuses on understanding users' security behaviors when interacting with systems, and developing secure, usable systems for legitimate users. Recent research has highlighted user anxiety related to the irreversibility of data exfiltration [22] and privacy concerns when storing data with cloud providers [8]. Cloudsweeper targets both of these concerns by providing additional security for sensitive data stored in the cloud.

Our system takes a different tack with respect to usability research: what are the usability concerns of attackers, and how can we make the system less usable in the presence of economically motivated attackers? Here usable security research investigating passwords and other authentication mechanisms can provide insights into the concerns of the attackers. One attacker goal might be to elevate their privilege by learning the answers to users' password reset and personal information questions for high value online accounts. Usability and security concerns regarding these types of authentication mechanisms have been investigated with respect to user cognitive limits when using such mechanisms [17, 18], users' choices when given the option to create their own challenge questions [10], and the security of personal information questions used as backup authentication when such personal information is commonly publicly available [1, 15].

2.2 Understanding attacker goals

In line with understanding the usability concerns of an attacker, one can also examine their desired outcomes to inform effective defenses. In most cases, cybercriminals do not compromise security systems without a reason, and do so instead with a specific goal in mind: to make money. Research exploring the monetization of botnets and spam have allowed the community to propose new strategies for preventing monetization, and thus decreasing the motivation to compromise security [11, 12]. The Judo system is one such example: by specifically targeting botnet-sourced spam based off of expressive templates, the Judo system creates a spam filter with a very low overall recall, but is extremely effective at denying spam sent by botnet operators [14]. By specifically targeting our protection to data that is useful to cybercriminals, we hope to both make compromising user accounts less financially lucrative, as well as lower the harm experienced by victims.

2.3 Mechanisms for data-centric security

Recent systems have been developed with a focus on protecting data outsourced to the cloud, even in the event of a system compromise. For instance, CLAMP implements a mechanism to prevent data leaks due to web server or database compromises by limiting the information accessible to an individual user via a trusted module [13].

Attack surface minimization, an example of which is introduced in Gondi et al. [7], acknowledges the liability inherent in keeping data after it is no longer useful: this system performs static analysis to determine the time of last use for stored information, and securely erases it immediately. As with many systems of this type, the authors treat all input data as sensitive to provide strong security guarantees. In this project, we wish to discern between potentially lucrative and inane. For these purposes, the RIFLE system provides a data protection mechanism that can be configured at runtime by users to suit their security needs [21]. Yumerefendi et al. describe a system that builds upon the concepts from RIFLE, and introduces a mechanism that helps users determine which data is sensitive through "doppleganger" processes executed alongside processes that handle sensitive information [24].

Other recently developed systems leverage the ability to expire data for improved security. Geambasu et al.'s Vanish system embeds cryptographic keys to stored data within a worldwide DHT, and controls the extent of the keys' replication to prevent the key from being recovered from the DHT after a configurable amount of time [5]. Geambasu et al.'s Keypad and Tang et al.'s CleanOS leverage data expiration to mitigate data compromise in the event of a stolen or compromised smartphone: through auditing filesystems and cloud-enabled cryptography, user data is kept safe while the usability of the device is not materially impacted [6, 20].

3. Cloudsweeper DESIGN

Cloudsweeper was designed to withstand full compromise of a user's cloud service credentials while presenting a straightforward interface. Here we describe both the threat model employed as well as the design goals for the system.

3.1 Threat model

The primary adversaries considered by the Cloudsweeper threat model are economically motivated cybercriminals and opportunistic eavesdroppers. We assume that the attacker has complete control over the cloud storage account in question, either through stealing the password or an already authenticated device. This assumption also includes the attacker having bypassed additional authentication methods, such as two factor authentication systems like RSA's SecurID or smartphone-based systems, as these systems have been previously successfully attacked. [2, 3].

Furthermore, our threat model is focused on data at rest rather than data in motion: Cloudsweeper cannot protect sensitive messages delivered to a user while their account is compromised. The main goal of Cloudsweeper is to presuppose an adversary with complete control of an individual's online archive, and develop solutions that mitigate the harm caused by such a compromise.

3.2 Design goals

Tolerate complete account compromise. The primary goal of Cloudsweeper is to protect the user in the event of a complete account compromise. We wish to maintain a user's data security even if their entire account is compromised.

Prioritize high-risk information. This goal encapsulates the insight that attackers do not break into accounts simply to compromise the security of a user; they most often have specific goals in mind. Only a very small proportion of the data in a given storage account is useful to attackers, so successfully defending only that data can preserve the user's security if their account is compromised.

Retain cloud storage advantages. Storing documents to the cloud has several advantages for the user, including increased reliability, accessibility, and computational power. Additionally, service providers use access to the plain text of a message to deduplicate storage, provide search services over the stored data, and serve contextual advertisements. A cloud security system should preserve these capabilities.

3.3 Heterogeneous documents

We propose heterogeneous documents as a solution for realizing the benefits of two competing strategies for securing online data archives without incurring the substantial costs associated with either strategy.

One approach to securing outsourced data is end–to–end encryption. This solution provides security benefits to the user by preventing attackers and eavesdroppers with access to the account from viewing any contained secrets. This "maximal storage security" strategy comes with downsides like losing the ability to outsource search or spam filtering to cloud systems.

Another approach to securing data stored in online archives is to make accessing the account more difficult. Two factor authentication is a common example of this strategy. The security these systems provide come with their own downsides, such as user inconvenience and difficulties with automated or legacy login systems. These "maximal perimeter security" strategies also provide no harm mitigation once an attacker has gained access to the online account.

Heterogeneous documents combines these approaches to protect data within an archive by storing encrypted sensitive information in-line with the rest of the document. Either the system or the user designates some portion of a document as sensitive; the system then encrypts the sensitive portion, saves the new version of the document within the cloud storage account, and deletes the original copy. The encryption key is not chosen by the user, but instead automatically generated for the user per Cloudsweeper session. This key is then necessary for decrypting the protected information.

By concentrating only on sensitive information, the goal is to encrypt information that has low "everyday" utility for the user, such that performing a decryption is rarely necessary. Additionally, by only encrypting the sensitive information, users will still be able to extract meaningful information from the context of the message, and will still be able to search for it by other keywords. The intuition here is that the vast majority of information stored within this account is of little or no use to an attacker; only a very small proportion of the information is of any use to him. Conversely, on a day to day basis the utility of lucrative information to its owner is likely very low. By making user effort with regards to security commensurate with the risk involved, we hope to make sensitive information much more secure, while imposing minimal additional effort on the user.

By emphasizing that the key be kept as a hard copy rather than saved on the device, we aim to drastically limit the number of attackers eligible to compromise the system: a successful attacker must gain access to the physical key storage location, or convince a user to scan his key. By limiting the utility of this key to accessing only sensitive, uncommonly used information, our conjecture is that Cloudsweeper can deter phishing; we leave exploration of this possibility to future work.

4. IMPLEMENTATION

The Cloudsweeper prototype is comprised of three components: a user interaction and authentication front end, a message translation and encryption engine, and a cloud storage communication back end.

Figure 1 presents an overview of our architecture. The functionality of each component is explained in detail below.

4.1 Front end

The user facing component of Cloudsweeper is implemented as an HTML5 web application using the Tornado web

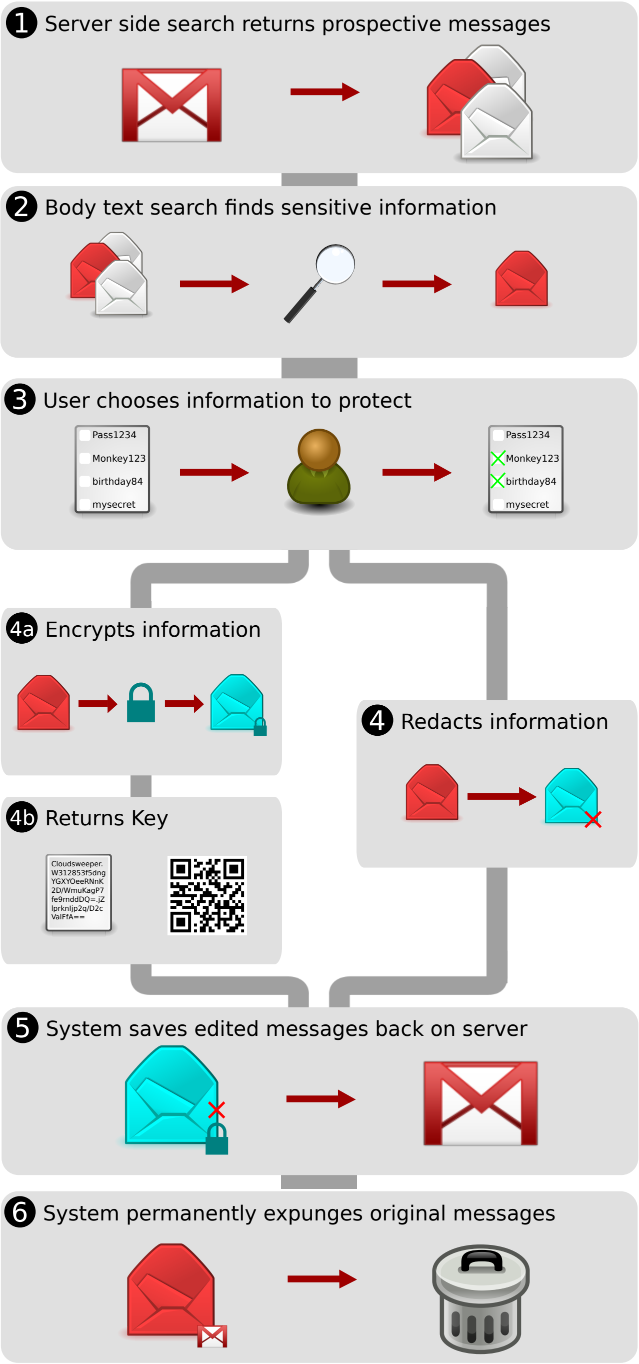
1 Server side search returns prospective messages
2 Body text search finds sensitive information
3 User chooses information to protect
Pass1234
Monkey123
birthday84
mysecret
Pass1234
Monkey123
birthday84
mysecret
4a Encrypts information
4 Redacts information
4b Returns Key
Cloudsweeper.
W312853f5dng
YGXYOeeRNnK
2D/WmuKagP7
fe9rnddDQ=.jZ
lprknljp2q/D2c
ValFfA==
5 System saves edited messages back on server
6 System permanently expunges original messages

Figure 1: Workflow diagram for Cloudsweeper. Only step 3 requires user interaction.

framework. When a user first visits Cloudsweeper to search for plaintext passwords, they are asked to give Cloudsweeper access to their Gmail accounts using OAuth, so that the user's Gmail credentials are never exposed to Cloudsweeper. As Cloudsweeper searches the account, found passwords are grouped together and displayed along with a count of the number of messages that contain the password and which email addresses sent a message containing the password. The list of found passwords is sorted in descending order of the number of different email addresses sending that password to the user. In our experience this order has a good chance of prioritizing user-selected passwords, due to password reuse among different providers.

The password search takes between several seconds to many minutes, depending on both the number of messages in the searched account and the number of messages containing plaintext passwords. Once the search is complete, users are able to redact or encrypt any or all of the found passwords.

4.1.1 Password redaction

If a user chooses to redact passwords from their account, they are warned that they will irreversibly modify the selected messages. Once they confirm, Cloudsweeper presents a progress bar while the cloud storage communication process modifies their messages. Once this process is complete, each redacted password is replaced with the token "[Cloudsweeper Removed]". The message is otherwise unchanged, and the original version is permanently expunged.

4.1.2 Password encryption

If the user instead chooses to encrypt their plaintext passwords, they are provided with a key that can be later used to reverse any passwords that have been encrypted. Keys are generated per user session, and persist until the user logs out or their session expires. All encryptions that a user performs during the same session will use the same key. The next time the user visits Cloudsweeper, they are assigned a new key to perform encryptions during the new session.

Cloudsweeperpresents the user's key in two forms, first as a Base64 encoded string and second as QR code. These representations are equivalent and either version of the key can be used for decryption. As with redaction, only passwords are replaced in each message. Cloudsweeperreplaces each password with a Base64 encoded ciphertext.

4.1.3 Password decryption

If a user wishes to reverse the encryption process, they can do so by visiting the decryption page in Cloudsweeper and entering the previously provided key. The key can either be entered manually or the user's browser can scan the key in as a QR code using a webcam. Cloudsweeperalso encourages users to store their keys in a purely offline medium, to further sever the connection between access to the user's email account and access to the encrypted sensitive information in the user's messages.

4.2 Message translation and encryption

Each email messages body is parsed to extract sensitive information. Currently Cloudsweeper implements two protection modes: *encryption*, which uses symmetric encryption to quickly and easily protect information in a reversible manner, and *redaction*, which permanently destroys the sensitive data.

4.2.1 Message parsing

Search, redacting and encrypting passwords in users' email accounts requires Cloudsweeper to correctly parse thousands of emails in accessed accounts. Unfortunately, many email clients, particularly older and more esoteric ones, send incorrectly formatted email. As a result, Cloudsweeper must deal with messages that are malformed in various ways, including:

- Missing expected header fields, such as *Message-Id* (a client generated unique identifier for each message) or *Date*.

- Missing or incorrectly declaring the character encoding scheme used for a message or message subsection (e.g. declaring that a UTF-8 encoded body section is actually ASCII).

- Improperly declaring the transfer encoding of the text in a body section (ex declaring that Base64 encoded text is really quoted-printable text).

Because of the sensitive nature of the data being operated on, along with the high amount of trust being asked for from users and the potentially high costs of data loss that could amount from improperly translating a message when rewriting it in a user's archive, Cloudsweeper is conservative when editing any email messages. If any messages are malformed according to RFC2822 [16], Cloudsweeper will not modify the message, and warn the user that some messages could not be properly parsed.

4.2.2 Safely rewriting messages with IMAP

The IMAP protocol does not provide the ability to edit stored messages, making it difficult to operate on and secure a user's email in place. To achieve editing-like functionality while preventing data loss in the event of a failure, Cloudsweeper uses a transactional system that creates a duplicate copy of the email in the user's Gmail account, deletes the original message, creates the new redacted or encrypted copy, and then deletes the backup copy. This ensures that message threading is properly preserved within the archive. If either the IMAP server or Cloudsweeper fail at any point during this process, no message is lost and the user is able to recover either the original or modified version of the message.

4.2.3 Encryption

Cloudsweeper uses AES256 in cipher block chaining mode to encrypt plain text passwords. Once a user authenticates with Cloudsweeper, their session is associated with a randomly generated 256 bit key and a randomly generated 128 bit nonce for the duration of their session. These values are used as follows when encrypting each password:

1. The password being encrypted is padded out with null bytes so that it cleanly divides into 16 byte blocks.

2. 16 bytes are randomly generated and saved as the Initialization Vector.

3. The padded password is encrypted under AES256-CBC using the generated IV and the user's key.

4. The IV is prepended to the ciphertext.

5. Both the user's nonce and the resulting ciphertext are Base64 encoded.

6. The password is replaced in the body of the email message with the following tag: [<nonce> <IV> <ciphertext>]. The format of this tag is designed to be easily searched for if the user wishes to decrypt this message, but not easily found without knowledge of the nonce.

The user is then provided with a separate Cloudsweeper key that they can use to decrypt all messages encrypted in this session. This Cloudsweeper key takes the following form <Base64 encoded key>.<Base64 encoded nonce>.

4.2.4 Decryption

Cloudsweeper also allows users to reverse the encryption of the plaintext passwords and restore the messages in their account to their original plaintext state. The user first submits the Cloudsweeper key provided at encryption time. Cloudsweeper then retrieves all messages that contain the nonce in the user's Gmail account.

For each returned message, Cloudsweeper then extracts the middle, encryption key from the provided Cloudsweeper key, Base64 decodes it, and uses it to reverse the encryption of each found ciphertext block with a matching nonce. Finally, Cloudsweeper saves the decrypted version of the message back into the user's email account.

4.3 Cloud storage communication

The service also communicates with the cloud based backing store. While currently implemented to communicate with Gmail via IMAP using Google's OAuth authentication API, Cloudsweeper can be implemented on top of any user-facing API for creating, reading, and deleting files or messages. Yahoo!, for example, provides similar OAuth access to their email accounts. Even without OAuth or IMAP access, messages could still be deleted and re-inserted via SMTP (although these forged messages might be subject to spam filtering).

4.4 User privacy

Cloudsweeper takes many steps to protect user's privacy and data. Most importantly, Cloudsweeper never stores encryption keys, found plaintext passwords, or any other sensitive or identifying user information beyond the lifetime of the user's session (user sessions either end when a user explicitly logs out, or four hours after the last user interaction). All user information is stored in memory and never saved in a persistant store or swapped out to disk.

Cloudsweeper further protects users' privacy by using Gmail's OAuth API for authentication. This allows users to give Cloudsweeper access to their email archive without needing to reveal their account credentials.

5. EVALUATION

To evaluate the Cloudsweeper system, we implemented a sample workflow focusing on finding passwords that have been emailed to the user in plaintext. For this approach, the server side search looks for the word "password." For the second pass search, we built a regular expression that extracts the expected password from the message based on contextual clues in the email. While straightforward, we believe that this strategy is sufficient for two reasons: firstly, "password" is often being used as a technical term and rarely if ever are synonyms used, especially in computer-generated template emails. Secondly, our goal is not to perfectly redact the archive but to sufficiently protect the archive such that an automated attacker with complete access would not be able to programmatically extract lucrative information. While we cannot prove impossibility, we believe that this redaction technique will materially improve the information security of a user while not impacting the utility of their cloud storage account.

To validate our assumptions regarding finding plaintext passwords, we used the corpus of 1,286 emails written in English which contain plaintext passwords posted at `http://plaintextoffenders.com` [19]. Of these, 1229 (96%) are successfully detected by our handwritten regular expression. A technology like Judo could be used as a more comprehensive solution [14] to efficiently create regular expressions to extract these terms.

5.1 Usage statistics

	Users	*Messages*
Search	1,992	170,609
Encrypt	497	19,664
Redact	179	5,444

Table 1: Usage statistics of opt-in research subjects for Cloudsweeper. The Users column indicates how many users performed a given action, and the Messages column indicates the total number of messages in which plaintext passwords were either found or acted upon by the user.

By default Cloudsweeper stores no information about users beyond temporary session data. Before a user performs any actions with Cloudsweeper, they can opt into allowing Cloudsweeper to record anonymous, non-sensitive information about their email for research purposes. Of the subset of users who have opted into this data collection, 1,992 Gmail accounts have been searched. 497 users chose to encrypt the results of their search, resulting in 19,658 messages encrypted using the heterogeneous documents approach. 179 users chose to redact their passwords in 5,444 messages.

5.2 Performance tests

Cloudsweeper's performance was evaluated by three different tests on a research subject's Gmail account which has been in use since May 2006, designated "test account" in the description below. The account contains approximately 100,000 messages, 4,812 of which contain the word "password", and 1,543 tokens that Cloudsweeper determined to potentially be passwords.

Each of the three tests were instrumented versions of standard Cloudsweeper functionality: search, encrypt and decrypt. The *Search* test searched the test account for the word "password", retrieved and searched the first test section of the matching messages from Gmail for tokens that were likely passwords using a regular expression, and then displayed information about each found password to the user.

The *Encrypt* test fetched 50 messages from the "test account" that contained plaintext passwords, deleted each message from the Gmail account, encrypted the plaintext passwords, and then saved the edited version of each message back into the "test account".

The final *Decrypt* test reversed the changes made by the *Encrypt* test. The "test account" was searched for the previously encrypted messages. The 50 messages containing encrypted passwords were fetched from Gmail, and the encrypted version of each message was deleted from the account. The passwords were decrypted in Cloudsweeper, the message text updated with the encrypted block replaced with the newly restored plaintext, and the updated message inserted back into the "test account".

	Search	*Encrypt*	*Decrypt*
Throughput (messages/min)	1902.0	3.6	7.2
IMAP (percent)	43.2%	99.3%	100.0%
Total execution (seconds)			
test account	152.0	783.0	412.0
average account	3.0	657.6	328.8

Table 2: Message search and modify throughput and elapsed time measurements for Cloudsweeper. IMAP (percent) indicates the proportion of time the Cloudsweeper server spent waiting for responses to IMAP commands from the Gmail server. The test account searched 4,812 messages and encrypted/decrypted 50 messages. The "average account" total execution time is synthesized based on these throughput measurements and the average number of messages searched/encrypted for the 497 users who chose to perform an encryption.

5.3 Performance measures

Table 2 provides an overview of Cloudsweeper's search, encryption, and decryption performance. Because a vast majority of the total execution time for modification actions is spent waiting for responses from the IMAP server, we do not present the execution time for redaction.

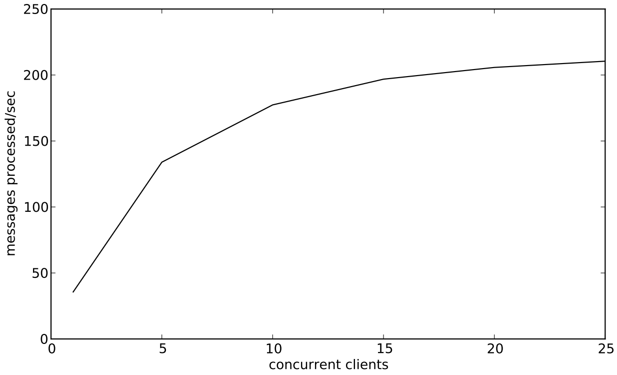

Figure 2: Message search throughput with multiple simultaneous clients.

5.4 Scalability

We tested Cloudsweeper's performance servicing many simultaneous clients within an environment of simulated users and Gmail servers. To simulate the Gmail servers, we instrumented the test account's communication with the live IMAP and OAuth servers to determine the latency distribution for requests to both of these services. The simulated services replicated the responses and latencies of the real servers so that several synthetic clients could be launched simultaneously without need for multiple real Gmail accounts.

Client traffic was generated through the use of scripted, headless browsers. Each client began at the Cloudsweeper home page, authenticated through the OAuth flow, and initiated a plaintext credential search through the Gmail account represented by the stored IMAP traffic. Once the search was completed, the client logged out and we recorded the amount of time taken between each client requesting the home page and successfully logging out.

Trials were run with 1, 5, 10, 15, 20 and 25 simultaneous clients conducting plaintext credential searches against identical simulated Gmail accounts. Figure 2 shows the aggregate number of messages Cloudsweeper was able to request and search per second. Because Tornado is an event-driven, single threaded architecture, CPU time is the main bottleneck on execution time. In the live deployment of Cloudsweeper, we deterministically multiplex eight independent instances behind an nginx proxy, which has been enough to service as many as 136 simultaneous users.

6. DISCUSSION AND FUTURE WORK

For privacy sensitive users, Cloudsweeper can also be implemented as a local service rather than a web service. When implemented as a web service, the user must trust the Cloudsweeper provider nearly as much as they trust their service provider. Deploying Cloudsweeper as a web service allows us to maintain a single codebase that works across all browsers, quickly fix deployed bugs, and makes data collection from research subjects faster and easier.

The threat model for this project focuses on illegitimate users who gain access to a user's account through public facing interfaces; attackers like rogue employees or state sponsored dragnets might be able to undelete messages redacted by Cloudsweeper, or store a copy of incoming messages before a user has a chance to modify them. We focus on public facing interfaces because Cloudsweeper is intended to primarily protect information at rest for extended periods of time from adversaries who have gained remote access capability through either malware or credential theft. We believe that this threat model is the most likely to apply here, and exposes the dichotomy between unfettered remote access and location- or artifact-limited (e.g. printed cryptography key) access.

One anticipated concern with the Cloudsweeper implementation is that a determined attacker with access to the account could use email-based password reset functionality to gain access to services even if their password reminder emails were protected. This issue is a side effect of the dual purpose that email accounts serve and not a problem with the heterogeneous documents strategy in general. Because modern cloud based email serves both a communication and archival purpose, an attacker could use a breached email account to reset the obscured credentials and generate new ones for the services they wish to access. This attack is specific to plaintext passwords in email, and does not apply to other online archives, e.g. Dropbox.

Additionally, while an attacker could use password resets to replace credentials secured by Cloudsweeper, securing an email account with Cloudsweeper still brings with it multiple security improvements, including securing credentials sent from parties that do not have password resets (friends, co-workers, etc.), preventing the attacker from harvesting credentials for reuse elsewhere, and securing plaintext passwords sent from accounts that are no longer active.

6.1 Future work

The current Cloudsweeper implementation exists as a proof of concept for the heterogeneous documents abstraction, and can be extended in many ways. Cryptographically, a public key scheme could be used, with the public portion kept at the server and the private portion kept as the offline token. This would allow reuse of an individual key to encrypt additional data without needing the private key present. With a public key scheme, automated periodic sweeps would also be possible.

Our current focus on passwords stored as plaintext is motivated by the intuition that such information is both considered sensitive by the user and lucrative to the attacker. Another key component in a comprehensive approach to attacker- focused security would be reconnaissance of attackers who seek to compromise systems and exfiltrate data. Recording what they search for and what they are able to successfully find would allow a future implementation of Cloudsweeper to more fully protect a user's account.

Extending Cloudsweeper to be compatible with different online services, notably cloud storage providers like Dropbox or Box.net would complement the current mail focused implementation. While OAuth has gained popularity as a mechanism for programmatically delegating authentication, the API for cloud storage accounts is not standardized. Thus, an implementation would be specific to each API, and the versioned nature of many cloud storage accounts may not allow the user to to fully expunge previous versions of a file. Our hypothesis is that different types of sensitive information might be kept in cloud storage style accounts than email accounts, and a Cloudsweeper implementation for these would be worthwhile.

Another straightforward extension is arbitrary redaction and encryption. Using an additional user interface within the email client, information within individual messages could be tagged for protection, or certain pieces of information marked as "sensitive" such that they are redacted whenever mentioned in an archive. A mechanism for securely recording these annotations would be required, as any system which stockpiles a record of lucrative data would be very lucrative to a cybercriminal.

7. CONCLUSION

Looking forward, the amount of data we record and store regarding our daily lives will only grow. Given the option, many people will choose to save something rather than discard it forever. Both because it is infeasible to manually manage such volumes and proactive user effort is rarely expended for security purposes, we need automated tools to assist in this task. Heterogeneous documents and Cloudsweeper exist as a proof of concept that it's possible to programmatically find and protect information that is more risky than it is useful. By focusing on automated strategies for finding sensitive information we can level the playing field with cybercriminals, thereby forcing determined attackers to conduct manual inspections of stolen accounts for "diamonds in the rough," which is unlikely to be a profitable venture.

8. ACKNOWLEDGEMENTS

We would like to thank Brian Krebs for his feedback and publicity of the Cloudsweeper system. This work was supported in part by the National Science Foundation under grant DGE-1069311.

References

[1] J. Bonneau, M. Just, and G. Matthews. What's in a name? evaluating statistical attacks on personal knowledge questions. In *Proceedings of the 17th International Conference on Financial Cryptography and Data Security*, 2010.

[2] A. Coviello. Open letter to rsa customers. `http://www.sec.gov/Archives/edgar/data/790070/000119312511070159/dex991.htm`, 2011.

[3] P. Crosman. New breed of banking malware hijacks text messages. `http://www.americanbanker.com/issues/178_111/new-breed-of-banking-malware-hijacks-text-messages-1059745-1.html`, 2013.

[4] M. Czerwinski, D. Gage, J. Gemmell, C. Marshall, M. Pérez-Quiñones, M. Skeels, and T. Catarci. Digital memories in an era of ubiquitous computing and abundant storage. *Communications of the ACM*, 49(1), 2006.

[5] R. Geambasu, T. Kohno, A. Levy, and H. Levy. Vanish: Increasing data privacy with self-destructing data. In *Proc. of the 18th USENIX Security Symposium*, 2009.

[6] R. Geambasu, J. John, S. Gribble, T. Kohno, and H. Levy. Keypad: an auditing file system for theft-prone devices. In *Proceedings of the sixth conference on Computer systems*, 2011.

[7] K. Gondi, P. Bisht, P. Venkatachari, A. Sistla, and V. Venkatakrishnan. Swipe: eager erasure of sensitive data in large scale systems software. In *Proceedings of the second ACM conference on Data and Application Security and Privacy*, 2012.

[8] I. Ion, N. Sachdeva, P. Kumaraguru, and S. Čapkun. Home is safer than the cloud!: privacy concerns for consumer cloud storage. In *Proceedings of the Seventh Symposium on Usable Privacy and Security*, 2011.

[9] W. Jones. *Keeping Found Things Found: The Study and Practice of Personal Information Management: The Study and Practice of Personal Information Management.* Morgan Kaufmann, 2007.

[10] M. Just and D. Aspinall. Personal choice and challenge questions: a security and usability assessment. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, page 8. ACM, 2009.

[11] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, V. Paxson, G. M. Voelker, and S. Savage. Spamalytics: an Empirical Analysis of Spam Marketing Conversion. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2008.

[12] K. Levchenko, A. Pitsillidis, N. Chachra, B. Enright, M. Félegyházi, C. Grier, T. Halvorson, C. Kanich, C. Kreibich, H. Liu, D. McCoy, N. Weaver, V. Paxson, G. M. Voelker, and S. Savage. Click Trajectories: End-to-End Analysis of the Spam Value Chain. In *Proceedings of the IEEE Symposium and Security and Privacy*, 2011.

[13] B. Parno, J. M. McCune, D. Wendlandt, D. G. Andersen, and A. Perrig. Clamp: Practical prevention of large-scale data leaks. In *Proceedings of the 30th IEEE Symposium on Security and Privacy*, 2009.

[14] A. Pitsillidis, K. Levchenko, C. Kreibich, C. Kanich, G. M. Voelker, V. Paxson, N. Weaver, and S. Savage. Botnet Judo: Fighting Spam with Itself. In *Proceedings of the Network and Diestributed System Security Symposium (NDSS)*, 2010.

[15] A. Rabkin. Personal knowledge questions for fallback authentication: security questions in the era of facebook. In *Proceedings of the 4th Symposium on Usable Privacy and Security*, 2008.

[16] P. Resnick. Internet Message Format, 2001. RFC 2822.

[17] S. Schechter and R. Reeder. 1 + 1 = you: Measuring the comprehensibility of metaphors for configuring backup authentication. In *Proceedings of the 5th Symposium on Usable Privacy and Security*. ACM, 2009.

[18] S. Schechter, A. Brush, and S. Egelman. It's no secret. measuring the security and reliability of authentication via "secret" questions. In *Proceedings of the 2009 IEEE Symposium on Security and Privacy*, 2009.

[19] I. Tabachnik and O. van Kloeten. Plain text offenders. `http://plaintextoffenders.com/`, 2013.

[20] Y. Tang, P. Ames, S. Bhamidipati, A. Bijlani, R. Geambasu, and N. Sarda. Cleanos: Limiting mobile data exposure with idle eviction. In *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation*, 2012.

[21] N. Vachharajani, M. Bridges, J. Chang, R. Rangan, G. Ottoni, J. Blome, G. Reis, M. Vachharajani, and D. August. Rifle: An architectural framework for user-centric information-flow security. In *Proceedings of the IEEE/ACM International Symposium on Microarchitecture*, 2004.

[22] Y. Wang, G. Norcie, S. Komanduri, A. Acquisti, P. Leon, and L. Cranor. I regretted the minute I pressed share: A qualitative study of regrets on facebook. In *Proceedings of the Seventh Symposium on Usable Privacy and Security*, 2011.

[23] S. Whittaker, V. Bellotti, and J. Gwizdka. Email in personal information management. *Communications of the ACM*, 49(1), 2006.

[24] A. Yumerefendi, B. Mickle, and L. Cox. Tightlip: Keeping applications from spilling the beans. In *Proceedings of the 4th USENIX Symposium on Networked Systems Design and Implementation*, 2007.