SSARES: Secure Searchable Automated Remote Email Storage*

Adam J. Aviv *(aja2002@columbia.edu)*, Michael E. Locasto *(locasto@cs.columbia.edu)*,
Shaya Potter *(spotter@cs.columbia.edu)*, and Angelos Keromytis *(angelos@cs.columbia.edu)*

Abstract

Many users store their email on a central remote email servers. While they gain the benefit of regular backups and remote access, they must trust the server to protect the confidentiality and integrity of their email. Since most email is stored in plaintext, a server compromise implies the compromise of the users' email. Although users can employ an encryption scheme like PGP, this protection is not a complete solution since the headers remain unencrypted, and it requires action on behalf of the sender.

We propose an alternative solution that begins with the server using public-key encryption for incoming email, including headers, body, and attachments. Doing so protects stored email from attackers but also prevents users from remotely searching their email (a feature which both IMAP and POP services permit). To solve this problem we present Secure Searchable Automated Remote Email Storage (SSARES), a novel system that completely encrypts incoming messages and allows the email to be searched securely. By using a combination of Identity Based Encryption and Bloom Filters, SSARES does not reveal any information about either search keywords or queries and remains largely transparent to both the email sender and receiver.

1 Introduction

Most email is both sent and stored in a plaintext format. During transmission, known encryption standards such as SSL can protect a message from eavesdroppers, but email "at rest" (stored on the mail server) remains at risk. Servers that store email and provide remote access to a mailbox must be trusted by the user to protect the email's contents; a compromised server also implies the compromise of the user's email. All unencrypted content would then be exposed to the attacker.

The contents of an email can be secured using a public-key encryption standard such as PGP. In this case, PGP preserves the headers of the email so that the message can be properly delivered. Consequently, an attacker who has access to the headers but not the content can still partially compromise the users' privacy by determining who the user is communicating with. PGP-style protection also relies on the correspondents actively employing the encryption, and unfortunately, the use of PGP is not widespread among the general public.

The first step toward a solution to this problem involves the construction of an email system that provides confidentiality and integrity protection without the direct interaction of the user. We can automate the process of encryption on the remote email server (thus making it transparent to the end user). By doing so, we can assure that, no matter who the sender is, the contents will be protected once it arrives at the server. The users' normal email practices do not need to change, nor do they need to convince their correspondents to change theirs.

Our ideal system would protect the entire message, headers included, on the remote server upon arrival of new email. This requirement implies that the server cannot access any content in the email once the message is encrypted. Consequently, it cannot perform searching as in current remote email systems like IMAP or POP. The search process could occur on the client side, but that would require extraneous processing and bandwidth, since every message must be transferred, decrypted, and then searched. If the client is working from a mobile device or has a large amount of email, this choice involves serious time delays.

A simple solution would be to use some sort of hash table to reference keywords within a message. A user would send a hash of the keyword, and the server would use the hash table to determine which messages match the request without the need to decrypt any messages nor know what the keyword is. But, an attacker would also have access to the hash table if the server is compromised and can perform a dictionary attack using known or guessed keywords that are relevant to the victim. The attacker could also watch the user perform searches live as the user requests emails, and perform a dictionary attack against the hash requests. Not only do the emails need protecting, but the searching technique does as well.

Our threat model focuses on an attacker who can break into the server and download the contents of the mailbox for offline analysis, or observe the system in action, watching how messages are matched to try and determine the contents. Of course, once the server becomes compromised, all newly arriving unencrypted messages are trivially exposed to the attacker.

To solve the problem of protecting email "at rest" and allowing for keyword searching while minimizing the information exposure, we present Secure Searchable Automated Remote Email Storage (SSARES). Our system completely encrypts incoming messages but also allows a user to search their email securely on remote servers without revealing any information about the keywords of the messages or the search queries. The system is built using a combination of Identity Based Encryption (IBE) and

*This work was partially supported by the National Science Foundation through Grant ITR CNS-04-26623.

Bloom Filters [1]. We discuss the design of SSARES in Section 2, and present a preliminary evaluation based on our prototype implementation in Section 3.

2 Design

SSARES is composed of three distinct parts. The first part handles newly arriving email, encrypting it on the mail server. The second part operates on the user side, and handles the formation of search requests to the server. The third part handles searching on the mail server. All parts make use of IBE and Bloom Filters.

The IBE encryption used by SSARES is called PEKS (Public-Key Encryption with Keyword Searching) [2]. PEKS uses a public-key style of encryption. New emails are completely encrypted and the keywords are encrypted using PEKS public-key encryption. The user can create a "trapdoor," an encryption form of a keyword the user wishes to search for, using the user's PEKS private-key and can relay the trapdoor to the server. The server can use this trapdoor and the encrypted keywords (stored as part of each encrypted message) to perform a test that will reveal if there is a match, and the matching (encrypted) messages can be returned to the user.

PEKS meets our goal of an automated and transparent process, but because each keyword is encrypted individually and thus must be tested individually, the speed of searching can become an issue. For example, a mailbox with 100 emails that average 450 keywords each would require 450,000 PEKS tests to perform an exhaustive search. To improve performance without compromising security, we combined PEKS encryption with Bloom Filter storage and probing. As a result, an encrypted message in SSARES, in addition to the list of PEKS-encrypted keywords, contains a Bloom Filter. When the user sends the trapdoor for each keyword, he also sends a query filter. On the server, the searching component first checks that the query filter matches the message's Bloom Filter before testing the PEKS.

Bloom Filters have a corresponding error value, or false-positive rate, which needs careful consideration. A filter with a low false-positive rate would be vulnerable to a dictionary attack that would reveal with high certainty to an attacker whether a keyword is contained in a particular message. To combat the dictionary attack, we choose to use filters with a high false-positive rate (25%) to reduce this certainty, but would still eliminate the majority of the messages without having to use PEKS, thus improving search speed. We call this filter an "error prone filter."

3 Evaluation and Results

We evaluated each of the three components of SSARES: email production, query generation, and search. We used a sample set of 100 emails from the Enron Data Set [3] (most of which were very small in size). All tests were run on a Linux RedHat Enterprise PC with a Pentium 3 CPU. To test email production, we converted 100 emails into SSARES form and measured the speed of the process and the size of the resulting (encrypted and keyword/Bloom Filter-augmented) messages. We recorded an average increase between the unencrypted and encrypted formats of 37 times, and an average speed of encryption of 17.2 seconds/message, with a maximum of 2 minutes. As expected, the key factor is the number of keywords contained in a message.

To test query production, we created test queries with various numbers of keywords. The speed and resulting size of the queries are well within reasonable standards, never taking more the 2 seconds to produce queries with a size max of 3 KB. Using the produced queries we searched the encrypted emails and measured the searching speed. The overall searching speeds were shown to be reasonable but not efficient enough for use. Search speeds ranged from 4 seconds to over a minute for a mailbox of just 100 emails. On the other hand, the error prone filters had a clear impact on searching speed. When a query failed the filter, we were able to eliminate the emails in under a second, and on average we were able to eliminate 76 of the 100 emails per search by using the filter. Without the error prone filter, it is clear that searches could have taken hours not minutes.

4 Conclusion and Future Work

We have presented SSARES, a novel system that can store email in encrypted form on the mail server while still permitting a user to search their email archive. SSARES does so without exposing information about the queries or the email contents (keywords). The combination of a Bloom Filter and PEKS encryption provides an automated and transparent process. Although the initial performance results for SSARES are encouraging, some work remains to improve the performance of the search procedure. In particular, we plan to investigate an optimal balance between speed and security for the false-positive rate of the filter. Another approach would be the use of a third layer of keyword organization between the filter and PEKS. For example, sorting the PEKS by the first letter of the (unencrypted) word could, with a well-distributed set of keywords, reduce per message searching speed by a factor of up to 26, but the usefulness of the extra information gained by an attacker needs further consideration.

References

[1] B.H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July 1970.

[2] D. Boneh, G.D. Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. *EUROCRYPT 2004*, LNCS 3027:506–522, 2004.

[3] <Enronmail.com>. Enron data set, 2004.