Simple Authentication for the Web

Timothy W. van der Horst and Kent E. Seamons
Internet Security Research Lab
Brigham Young University
{timv,seamons}@cs.byu.edu

***Abstract*—Automated email-based password reestablishment (EBPR) is an efficient, cost-effective means to deal with forgotten passwords. In this technique, email providers authenticate users on behalf of web sites. This method works because web sites *trust* email providers to deliver messages to their intended recipients. Simple Authentication for the Web (SAW) improves upon this basic approach to user authentication to create an alternative to password-based logins. SAW: 1) Removes the setup and management costs of passwords at EBPR-enabled sites; 2) Provides single sign-on without a specialized identity provider; 3) Thwarts passive attacks and raises the bar for active attacks; 4) Enables easy, secure sharing and collaboration without passwords; 5) Provides intuitive delegation and revocation of authority; and 6) Facilitates client-side auditing.**

I. Introduction

Password logins are overused. Reusing a password across all web sites is risky, but managing multiple passwords results in frequently forgotten passwords. Many web sites handle forgotten passwords by emailing the user a password or a hyperlink to a facility to reset the password, a technique we refer to as email-based password reestablishment (EBPR).

This paper introduces Simple Authentication for the Web (SAW), a new web site login approach that improves both the security and the convenience of EBPR. SAW utilizes email for all authentications and not just for recovering from forgotten passwords. SAW also provides single sign-on and can be fully automated so that the login details are transparent to users.

A. Alternatives to Passwords

Password managers, such as Password Safe [1] and those built into popular web browsers, solve the problems associated with multiple passwords by remembering users' passwords. A single password is then used to protect the password manager. Nevertheless, password managers generally lack portability and require significant account-specific maintenance.

Other alternatives to passwords (see Section VII) require a specialized identity provider. As is evidenced by their lack of widespread adoption amongst web sites, finding a mutually trusted identity provider is difficult.

In 2003, Garfinkel [2] coined the term email-based identification and authentication (EBIA) to describe the general concept of using an email address as an identifier and the ability to receive email messages sent to that address as an authenticator. In evaluating this current trend, Garfinkel argued that EBIA's widespread use is evidence that the risks of this system are manageable, especially given that the alternatives are prohibitively expensive for many web sites.

The remainder of this paper is organized as follows. Section II provides an in-depth look at the goals and design of SAW. In Section III, we discuss a prototype implementation of SAW for web site logins. Section IV presents a detailed evaluation of the threats to this system. Section V presents some advanced features of SAW. Section VI shows how SAW overcomes the obstacles facing email-based authentication. Section VII discusses related work. Section VIII contains conclusions and a discussion of our future work.

II. SAW

As password-based, EBPR-enabled sites already demonstrate their willingness to offload user authentication to email providers, the presentation and evaluation of the SAW protocol focuses on deployment to web sites inclined to assume the risks of EBPR. Sites reluctant to accept these risks (e.g., online banks) can still benefit from SAW by deploying it as an additional factor of authentication.

For simplicity, this paper defines a secure login as one that uses HTTPS and an insecure login as one that does not. We acknowledge the existence of password-based authentication mechanisms (e.g., SRP [3], DH-EKE[4]) that securely operate over insecure channels, however use of these protocols by web sites is rare. Both secure and insecure logins are widely used and would benefit from the adoption of SAW.

A. Obstacles to Adoption

At EBPR-enabled web sites, users prove their identity by using their password or by demonstrating ownership of their email address. If the ability to receive email messages is sufficient to circumvent users' passwords, why not make email the primary means of authentication and remove site-specific passwords? We identify four obstacles:

1) Latency: In some cases, email message delivery and retrieval may require a relatively long period of time.

2) Lack of privacy: Email messages are typically sent without cryptographic protection and are therefore susceptible to passive eavesdropping and active modification.

3) Convenience: Password-based systems are pervasive and accepted by both users and web sites. Changing a web site's login system often requires significant time and resources as well as additional user training.

4) Reliance on a third party: By involving an email provider in the authentication process, a dependency upon a third party is introduced. If the email provider is unavailable, the authentication process cannot succeed.

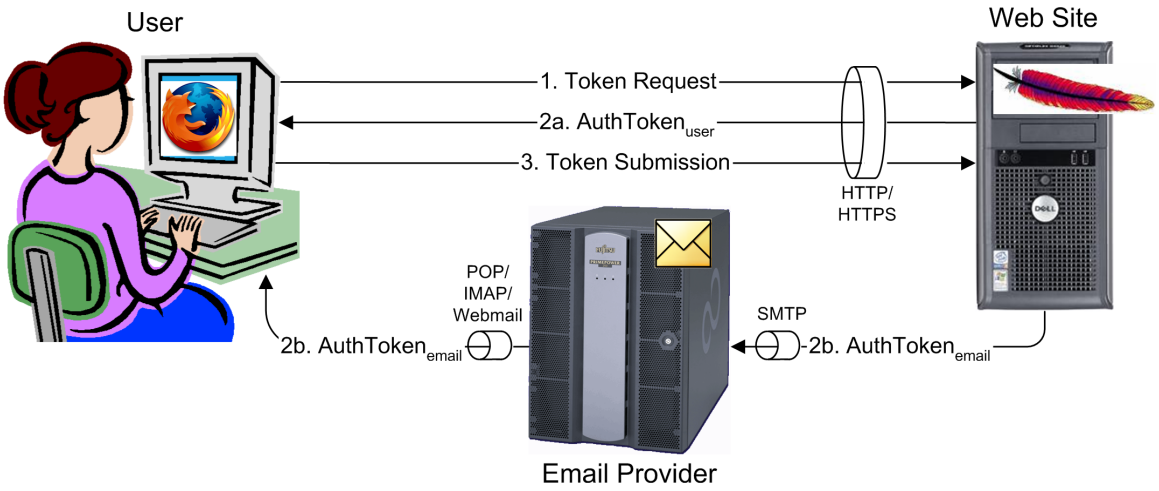

Fig. 1. The SAW protocol. Based on the user's email address, specified in (1), a web site creates and distributes two authentication tokens. AuthToken$_{user}$ (2a) is sent directly the user as an HTTP cookie. AuthToken$_{email}$ (2b) is emailed to the user. Both tokens must be returned to the web site (3) to successfully authenticate. Each web site login attempt involves its own unique, short-lived, single-use tokens.

B. Goals

The design goals for SAW are:

- Remove the need for users and web sites to setup and manage passwords
- Provide web single sign-on via email
- Require no modification to email providers
- Thwart all passive attacks
- Raise the bar for active attacks
- Overcome the obstacles identified in Section II-A
- Reduce user involvement through automation to make logins more convenient and reduce the attack surface for phishing and social engineering attacks
- Offer advanced features usually unavailable in password-based systems
  - Easy, secure sharing and collaboration without passwords
  - Intuitive delegation and revocation of authority
  - Client-side auditing

The remainder of this section describes SAW's improvements in security over existing email-based authentication systems. In-depth discussion of the advanced features of this system and how it addresses all of the obstacles listed above is left to Sections V and VI, respectively.

C. Protocol

The steps to authenticate using SAW (see Figure 1) are as follows:

1) Token Request: The user submits (e.g., via a webform) her email address to web site in the *Token Request* message. This request *may* be sent over HTTPS (see Section IV-A).

2) Token Response: The web site, based on the permissions of the email address, creates several short-lived, single-use *Authentication Tokens*, hereafter referred to as *AuthToken*$_x$, where x is the identifier of the specific token. These tokens are created based on a security parameter k.

If the address is authorized, AuthToken$_{complete}$ is chosen at random from $\{0,1\}^k$, and split into two shares as follows:

$$AuthToken_{user} = AuthToken_{email} \oplus AuthToken_{complete}$$

where AuthToken$_{email}$ is another random value from $\{0,1\}^k$. This method of token creation is a conventional secret splitting scheme [5] and provides the perfect security of one-time pad encryption. No amount of computation will recover one share without the other two.

AuthToken$_{complete}$ is cached by the web site in a temporary look-up table. AuthToken$_{user}$ is returned directly to the user as an HTTP cookie. An email message containing AuthToken$_{email}$, and the identity of the web site that created it, is sent to the user's email address.

If the address is not authorized, a random member of $\{0,1\}^k$ is returned as AuthToken$_{user}$, and, in lieu of AuthToken$_{email}$, a human readable explanation of the failure is emailed. Always returning an AuthToken$_{user}$ prevents an impersonator from learning anything about the email address owner's permissions. This is reminiscent of password systems that do not disclose that a username is invalid.

3) Token Submission: The user retrieves the email message sent by the web site, extracts AuthToken$_{email}$, and returns both tokens to the site. If these values combine to equal the AuthToken$_{complete}$ for that particular user and token identifier (see Section II-D), then the authentication is successful and the system uses a session-level trust preservation mechanism for the remainder of the session (e.g., a session cookie).

D. Token Identifiers

Each set of tokens is given a unique identifier, $transID$, to facilitate the matching of an AuthToken$_{user}$ with its corresponding AuthToken$_{email}$ and to enable web sites to look up the correct AuthToken$_{complete}$. As every Token Request results in the creation of a new set of tokens, this identifier allows a user to have multiple concurrent authentications in progress

with the same web site, thus permitting a user to retry the authentication process without invalidating a previous attempt. Appending $transID$ to the *name* of the HTTP cookie used to convey AuthToken$_{user}$ prevents the cookies from current authentication attempts from being overwritten.

E. Client-side Automation

Manually polling an email account for a specific message is inconvenient and unnecessary. Additionally, a carefully crafted phishing email resembling a token message could lure an unsuspecting user to a malicious site.

Automating the process of retrieving and submitting AuthToken$_{email}$ enhances the convenience and security of SAW. The user's software agent verifies a token by checking its sender and identifier with its list of outstanding authentication attempts. It then submits the tokens to the desired site. Although a user could perform these same tasks, the user's software agent is able to do it much faster and more accurately.

Ideally, web browsers would have native support for SAW to allow single sign-on to all SAW-enabled web sites while the browser is open by having the browser cache the email account password for a limited time. Section III presents a prototype browser toolbar that implements these benefits.

F. Alternatives to Email

SAW provides personal messaging-based authentication. AuthToken$_{email}$ is easily delivered over a variety of personal messaging mediums (e.g., instant and text messaging). SAW provides a platform to explore the potential that these personal messaging systems or a hybrid combination of these mediums have for authentication.

Instant messaging, in addition to providing an attractive, low latency alternative for delivering AuthToken$_{email}$, also facilitates the use of SAW by those who rely on free web mail accounts (e.g., Hotmail, Yahoo! Mail, Gmail) since the programmatic access (i.e., POP/IMAP) to these accounts necessary for client-side automation is often only available to "premium" accounts. As these providers associate free instant messenger accounts (e.g., MSN Messenger, Yahoo! Messenger, Google Talk) with each email address, users are able to leverage the same password as their email account to enjoy the benefits of SAW and client-side automation.

III. Implementation

There is a myriad of web sites that are ideal candidates for adopting SAW. These systems include: blogs, e-commerce sites, photo sharing sites, digital libraries, forums, conference program committee sites, private wikis, mailing lists, and personal web sites.

SAW provides several advanced features (see Section V) that make it an even more attractive alternative to passwords. For example, since SAW provides decentralized authentication of users, it enables web sites to specify permissions for users outside its local security domain. This is powerful tool for sharing and collaboration (see Section V-A).

From: SAW_TokenGenerator@securecomm.org
To: TestSubject@some.edu
Subject: [SAW-https://securecomm.org/login] TransID=fc9f7de...

This email can be used in one of two ways:
If you are using the SAW toolbar:
This message will be handled and deleted automatically.

If you are NOT using the SAW toolbar:
Click on the link below ONLY if you recently initiated a request to log in to https://securecomm.org/login:
https://securecomm.org/login?TransID=fc9f7de2d1f6e5f4f93fa42ffaa9c13151450ffd4e1f73786346fed86e84d851&ATemail=2fe31bf9c5d91e3d2ebc7688c812ba51d203ef6cfda5d99cbe7f5472b8e3ad57

Fig. 2. An email message with AuthToken$_{email}$. The hyperlink is a convenient means for users who without the SAW toolbar to return the token to the site that created it and prevent inadvertent token disclosure to phishing sites.

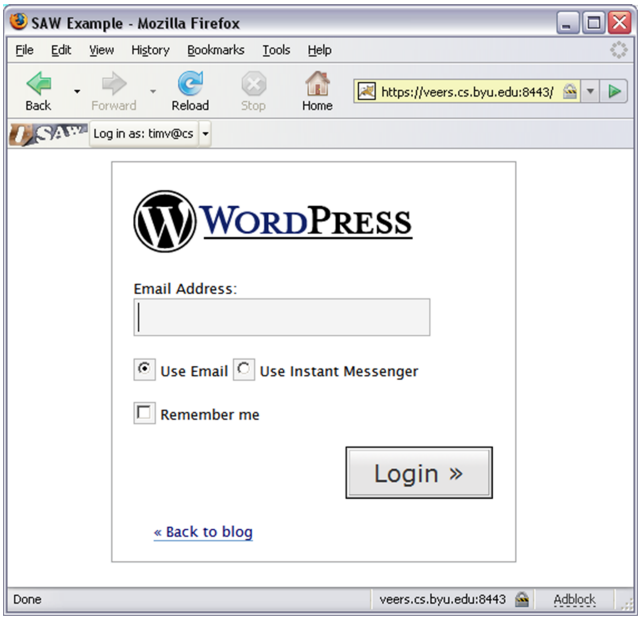

Fig. 3. A login page and the SAW toolbar. Clicking the toolbar's login button submits the selected email address, retrieves the email sent by the web site, and submits both authentication tokens. Without the toolbar these actions must be performed manually. (Note that only an email addresses is required.)

A. Blog Access Without Passwords

We added support for SAW to Wordpress [6], a popular web log platform, by creating a plug-in. Based on email addresses supplied by the login page, this plug-in creates and distributes the authentication tokens and then displays a page containing instructions to complete the authentication. With the client-side toolbar, the remainder of the login process is automated.

Without the client-side toolbar, the user must authenticate to the email provider and retrieve the message containing AuthToken$_{email}$ (see Figure 2). To facilitate the transfer of this token to the web site, this message includes a hyperlink that contains the token. Clicking on the link submits AuthToken$_{email}$ along with AuthToken$_{user}$ since it is a cookie set by that site. Current browser designs protect cookies

	Time to check for messages	Total login time using email	Total login time using IM
Gmail account	2.5 sec	4 sec	<1 sec
Private account	1 sec	1.5 sec	n/a

TABLE I
INITIAL PERFORMANCE RESULTS USING A HIGH-TRAFFIC, FREE WEBMAIL ACCOUNT (GMAIL) AND A LOWER-TRAFFIC, PRIVATE EMAIL ACCOUNT.

from being sent to any unintended site, thus eliminating the possibility of submitting AuthToken$_{user}$ to a phishing site.

The hyperlink in the email message exists as a convenience to users without a client-side automation tool. When such a tool is used, the entire message body can be dropped because all required information is contained in the subject line.

The client-side toolbar (see Figure 3) is implemented for both Internet Explorer and Mozilla Firefox. These toolbars share a common service that manages email account information and communicates with email providers via POP or IMAP. It can also receive instant messages using XMPP [7].

These toolbars provide a recognizable, uniform interface for authenticating to SAW-enabled web sites. They detect sites that accept SAW and provide a "Log in" button that enables one-click authentication to the web site.

Clicking on this button prompts the user for the email account password (once per session), submits the desired email address/IM handle to that site and begins checking the specified email, or IM, account for the message containing AuthToken$_{email}$. By remembering the email account password, the toolbar provides transparent single sign-on to all SAW-enabled sites for the current session.

B. Performance

An initial performance analysis shows promising results (see Table I). Before examining specific numbers, it is important to note that the time required to authenticate in SAW includes not only the time it takes for an email message to be delivered to an email provider, but also the time required to retrieve it from the user's mailbox. This analysis used two different email accounts and results are an average of 50 iterations.

The first account is a high-traffic, free webmail address provided by Gmail. This account is accessed using POP over TLS and requires 2.5 seconds for the toolbar to connect and retrieve an email. It takes 4 seconds from the moment the user clicks the log in button until authentication completes.

The second account is a lower-traffic, private email address that is also accessed via POP over TLS. This account requires one second to connect and retrieve an email and takes 1.5 seconds for completion of the entire login process.

When instant messaging is used (only the Gmail account provides IM capability) the entire login process takes less than a second.

IV. THREAT ANALYSIS

This section analyzes the threats and attacks to SAW. First, it considers the affects of passive eavesdropping on this protocol's communication channels. Next, it examines the ability of an attacker to actively impersonate a valid user. Then, it shows how this system mitigates the risks of password phishing, denial of service attacks, and spam. Finally, it examines the threats to login information stored at the web site, user, and email provider and how to address stale email addresses.

A. Passive Eavesdropping

Current EBPR methods are vulnerable to passive attacks. By emailing plaintext passwords, or a link to facilitate a password reset, anyone sniffing the network will be able to compromise the user's account. This section discusses the protections SAW required for each of its three communication channels:

1) User and Web Site: Although HTTPS provides confidentiality and integrity to the login process as well as authentication of the web site, a significant number of EBPR-enabled sites lack this protection. If an insecure channel is used for communication between the user and the web site, then the submission of a user's email address and the contents of the Token Submission message are passively observable.

Under these conditions, SAW closely resembles a one-time password system; authentication tokens become worthless to an attacker once users have submitted them to the site. SAW has the added benefit that these tokens must be used within a short time frame. This is a significant improvement over EBPR-enabled web sites with insecure login pages.

Note that without an HTTPS connection between the user and the web site, it is possible for an eavesdropper to see everything users see when they interact with the site and it may be possible for an attacker to hijack the user's session.

Also, without HTTPS on this link SAW cannot detect or prevent an active man-in-the-middle (e.g., an attacker with a spoofed domain name) and a replay attack would be possible.

2) Web site and Email Provider: Ideally, a secure channel should be also used for the communication between web sites and email providers, however, because the vast majority of email traffic currently does not have any cryptographic protections in place, it is not feasible, at this time, to make this a requirement.

Fortunately, the AuthToken$_{email}$ that is sent over this link is useless without the AuthToken$_{user}$ that was delivered directly to the user. When an HTTPS connection is employed between the user and the web site AuthToken$_{user}$ is never visible to an eavesdropper and the risks of sending AuthToken$_{email}$ in the clear are mitigated.

3) Email provider and User: The link between the email provider and the user requires a connection that provides confidentiality, integrity, and authentication of the email provider (e.g., POP/IMAP over TLS). This prevents local eavesdroppers from collecting the email account password.

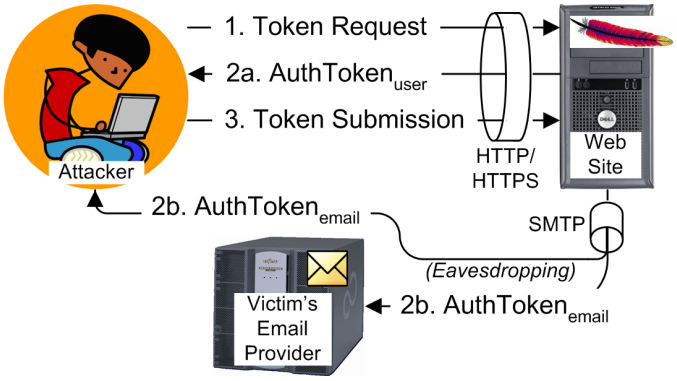

Fig. 4. In an active impersonation, the attacker submits the victim's email address in a Token Request (1). AuthToken$_{user}$ (2a) is returned directly to the attacker and, by eavesdropping the communication between the web site and the victim's email provider, the attacker obtains the corresponding AuthToken$_{email}$ (2b). The attacker is now able to impersonate the victim to that web site (3). EBPR-enabled sites are also vulnerable to a similar attack.

B. Active Impersonation

Provided at least one token is securely sent to the user, SAW eliminates the potential for passive attack. Nevertheless, attackers able to monitor the communications between the web site and users' email providers are positioned to mount an *active impersonation* (see Figure 4).

By submitting a Token Request, an active attacker obtains a valid AuthToken$_{user}$. By eavesdropping on the unprotected link between the web site and the email provider it is also possible for the attacker to intercept the corresponding AuthToken$_{email}$. Email providers, or more likely malicious insiders, are also able actively impersonate their users.

We believe that SAW is workable and usable, even in light of this attack. Sites that employ EBPR are also susceptible to a similar of attack in which an attacker requests a password reset for the victim's account and then eavesdrops the resulting email message sent by the web site. Nevertheless, as Garfinkel [2] states, the widespread adoption of EBPR indicates that these risks are manageable.

This potential for active impersonation brings to light a problem that SAW shares with PKI-based systems. Both systems are vulnerable to malicious or compromised trusted third parties; a certificate authority (CA) or email provider has the ability to undetectably impersonate its clients. Just as PKI-based system is only as strong as its CAs, SAW is only as strong as its email providers.

The ability to deliver email to only its intended recipients is a required characteristic for email providers who want to retain their users. Some email providers will be more desirable than others in the same way that some CAs are more trusted than others because of their reliability and privacy practices.

Section V-D describes several optional mechanisms that will prevent active impersonations by email providers and make it more difficult for an external entity to mount this attack.

It is possible to tunnel SMTP through a TLS connection; however it is not a common practice to do so. This is most likely due to the overhead involved in creating and accepting a large number of TLS connections. If, however, TLS was used to transfer messages between email providers this would eliminate the ability for an external entity to perform an active impersonation. Unfortunately, it does not thwart a malicious or compromised email provider from implementing this attack[1].

Tunneling SMTP through a TLS connection reduces the attack surface for active impersonation attacks, raising the bar significantly compared to current EBPR practices. However, current EBPR practices indicate that SAW is already workable and usable at many web sites without this advanced protection. Requiring secure email delivery strengthens SAW and could make it attractive to web sites that require stronger guarantees against active impersonation.

C. Password Phishing and Denial of Service

Phishing lures potential victims into divulging sensitive information (e.g., passwords) by emailing an official-looking message containing a link to a site that looks like a web site used by the victim. Since SAW eliminates site-specific passwords, the only thing a phisher gains from tricking users into attempting to authenticate a phishing site is an email address, which the attacker already knows because she sent the phishing message to it.

Unfortunately, although SAW protects users' ability to authenticate to specific web sites from phishers, it does not prevent an attacker from tricking users into believing that they have successfully authenticated and are now interacting with the real web site. This remains a difficult open problem, not just for SAW, but for web site logins in general.

As clicking links found in email messages is potentially dangerous, such links should only be followed when they are the result of a Token Request. The short time span between requests and delivery aids users in following this practice. Using hyperlinks is more secure than having the user cut-and-paste the token because it ensures that the token will be returned to the web site that issued it.

A race condition is possible when emailing hyperlinks to users. If an attacker is able to email a spoofed message before the web site's email message arrives, the attacker may be able to fool users into following a link to a malicious site. Although the users visit the attacker's site, none of the authentication tokens are disclosed to it.

Unfortunately, a quick visit to a malicious web site can be quite dangerous due to browser vulnerabilities, or sly social engineering, that result in the installation of malicious software. This is a very real threat and demonstrates the advantage of taking the user out of the loop and allowing these actions to be performed by the toolbar, which employs strict token matching to remove this race condition.

There are several ways to mitigate denial of service attacks. For example, a web site can limit the number of messages that it sends out to a single address. Also, client-side software can recognize unsolicited authentication emails as spam and automatically discard them or file them away for later analysis.

[1]Splitting AuthToken$_{email}$ amongst multiple email will prevent non-colluding email providers from mounting this attack (see Section V-D).

Also, it is possible for the token messages themselves to be categorized as spam. Garfinkel [2] suggests that emails used for authentication purposes should be signed by the sender to prevent this. This approach, however, may be too expensive for many web sites. In SAW, token messages are very easily identified and it is trivial to teach a spam filter that they are not spam. Any unrequested messages that do get through can be handled by the client-side software as stated above.

D. Storage of Login Information

In SAW, there is no long-term storage of sensitive user-specific authentication information at web sites. Although values for $\text{AuthToken}_{complete}$ are cached for users currently attempting to authenticate, these values are short-lived and are only used once.

There is also no long-term storage of authentication information by users. Although client-side automation tools may store email account information, the account password should never be stored outside of a session.

It is probable that email providers will maintain a long-term storage of messages containing an AuthToken_{email}. Fortunately, this token is useless without its corresponding AuthToken_{user} and both tokens are short-lived and single-use. This mitigates the risks of storing AuthToken_{email} at the email provider for auditing purposes. Offline analysis of the individual tokens yields no new information because each $\text{AuthToken}_{complete}$ is as likely as the next. Token size, determined by k, should be chosen to be sufficiently large to thwart any brute force attempt to guess them. Online, brute force attacks for the two tokens are trivially detected.

E. Stale Email Addresses

Allowing users to change their primary address at a web site is easy when their original email addresses are still accessible. When doing so, it is important that this process require an authentication challenge to the original address. A similar concept (requiring a user to enter the original password before it is changed) exists in many password-based systems.

A *stale email address* is an email address that has become inaccessible to the user. One approach to address this problem is to have a password backup or secret question/answer. These methods switch the traditional roles of emails and passwords as primary and secondary authentication mechanisms. Lamentably, this method negates many of the benefits of SAW, as it reintroduces passwords. We identify two approaches to deal with this problem.

First, if an email address is no longer accessible, a user can use whatever out-of-band techniques that were used to create the account at the web site in the first place. For example, if the email address of a member of a conference program committee goes stale, that member can contact the committee chair and have the email address changed to a new one.

Systems with open user registration usually do not have an out-of-band means to manage accounts. The simplest solution is to create a new account. However, it may be the case that the user has information stored at the web site under the old account, e.g., a collection of photos. In this situation, a secondary email address, preferably in a completely different domain than the primary one, registered with the web site during account creation presents an elegant solution for the vast majority of users. This method also mitigates the problem of relying on a single third party at authentication time by providing a "backup" that can be used if the primary email provider is currently unavailable.

V. Advanced Features

Currently, email-based authentication is predominantly relegated to automated password reestablishment. By extending and significantly improving this type of authentication, SAW brings it to the forefront and demonstrates its innate power as a primary authentication mechanism and as a viable alternative to passwords at EBPR-enabled web sites.

A major benefit of SAW is that email addresses, unlike digital certificates, are very easy to obtain and are useful in a variety of different applications in a myriad of security domains. It is also easier for a web site to trust almost any email provider because the amount of trust placed in them is much less than that of a traditional CA. Correspondingly, the implicit trust that email providers will only deliver an email message to its intended recipient allows them to be completely oblivious to the fact that they are performing the duties of an identity provider. This enables web sites to unilaterally deploy this authentication mechanism without modifying the email provider or entering into any legal or business relationships with them. Finally, in its simplest form, it requires no special user software.

A. Sharing and Collaboration

Users need a secure way to share files (e.g., photos, documents) with friends and collaborators. There is no easy-to-use, universal mechanism available to do this. To securely share files in password-based systems, each participant must have a username and password. The account creation process necessary for this mechanism is commonly a source of frustration.

Email messages are also commonly used to transfer and share files. We call this approach *informal sharing* because it does not require the recipient to possess a local account like the password-based approach. This method is easy, intuitive, and there are reasonable assurances that files will reach their destination. Emailing content is a *push-based* approach and due to the informal nature of the transaction, it is unlikely that the content will be sent with any cryptographic protection. Also, files sent via email can clog the recipient's mailbox.

SAW provides the foundation to build a secure, *pull-based* informal sharing mechanism. Since email addresses are necessarily unique they serve quite effectively as unique identifiers in access control lists. Using SAW it is possible to securely and efficiently use proof of email address ownership as an authenticator. Sharing, or allowing collaborative access, is accomplished by specifying the email address, or list of email addresses, that are permitted to access a resource. We call this approach *email-based access control* (EBAC).

AuthType Digest **AuthName** “private” **AuthDigestFile** /pwFile **Require user** alice bob	**AuthType** EBAC **AuthName** “private” **Require user** alice@a.edu bob@gmail.com
(a)	(b)

Fig. 5. A traditional .htaccess file (a) restricts access to a directory to set of known users, e.g., alice and bob. These users, and their passwords, are defined in the AuthDigestFile. With EBAC (b), this password file is eliminated and users prove ownership of their email addresses using SAW to gain access to a directory. This removes the need to create/distribute user-specific passwords.

This method of specifying permissions is simple and easy to use. It is also desirable for systems with discretionary access control (DAC), where users are allowed to specify access control for the content they control, e.g., photo sharing applications. These users can now securely share content, e.g., private photo albums, with their friends and family without requiring them to create accounts with the site or to manage long-lived links. This method is also well suited for use with the Apache Web server's .htaccess files, which permit user-specified, directory-level access control (see Figure 5).

We have implemented a module for Apache that provides this functionality and currently use it to protect a private collaborative wiki. In addition to enabling secure access control without having to create user accounts and distribute passwords, this module allows us to use SAW to protect access to an application without having to modify that application.

B. Delegation

In general, delegation of authority in password-based systems is accomplished by sharing the password. To the eyes of the system, the delegate is the delegator.

There are several problems with this approach. First, once a password has been given away, it cannot be revoked, with a reasonable degree of certainty, without changing it. By giving the delegate the same password as the delegator, too much authority is delegated. The delegate now has the power to change the password and revoke access from the delegator, at least temporarily. The delegate can also share the password with others without the approval of the original delegator. Often times it is hard to remember who has received the delegated password and why.

A key benefit to delegation in password-based systems is that the site does not need to be aware of the delegation. We call this capability *client-based delegation*.

Client-based delegation in SAW leverages email forwarding rules. A delegate provides the delegator's email address to the web site and, through a forwarding rule at the delegator's email provider, AuthToken$_{email}$ is sent to the delegate's email address. By removing this forwarding rule, a delegator immediately revokes permission for future authentications by a delegate. The delegator's list of forwarding rules provides an up-to-date record of delegations, which facilitates delegation management.

Client-based delegation in SAW does not require support from the web site. This type of delegation described works

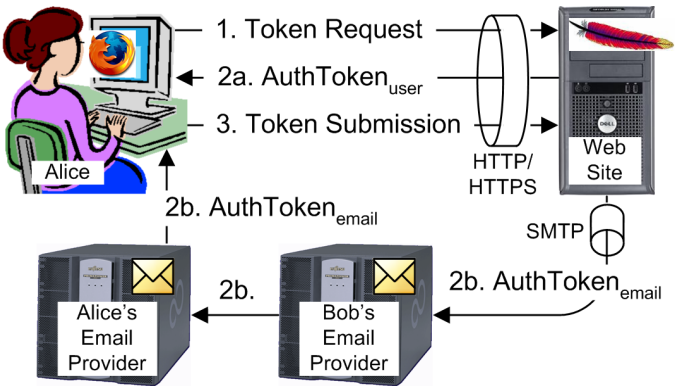

Fig. 6. Delegation SAW using email forwarding rules. Bob delegates access to a web site to Alice by creating a forwarding rule with his email provider. To access the site, Alice submits Bob's email (1). She then receives AuthToken$_{user}$ directly from the site (2a). Once AuthToken$_{email}$ (2b) arrives at Bob's email provider, the forwarding rule sends this message to Alice. Once Alice receives the message, she completes the authentication process (3).

best when the user's email provider enforces the forwarding rules. Forwarding rules specified and processed in a local email client (e.g., Outlook Express) are problematic because the delegate must rely on the delegator to retrieve the email message from the server before the client forwards it to the delegate. Although web site support for delegation is not required, such support could provide more control over which permissions are delegated and enable users that lack access to email forwarding facilities at their email provider to delegate their authority.

Various types of delegation are achievable through this technique. The simplest form is *complete delegation*, where all token messages are forwarded to the account of the delegate. This can also be accomplished by specifying the delegate's email address as the delegator's secondary email address, but this method is not as desirable as the former because it requires modification to data stored at the web site and is not as scalable. In this model, it should be possible for the delegate to change the primary/secondary email accounts associated with the site. This form of delegation is helpful for users with multiple email addresses.

Multiple email addresses significantly improve the privacy of the user. For example, by using a free email address as opposed to a more identifying address, e.g., a business email, users avoid leaking information about themselves, e.g., their place of employment or real name. Using multiple addresses for authentication also limits the information that colluding web sites glean from pooling their information. Using the complete delegation model the token requests for multiple accounts funnel into a single one, enabling the user to authenticate to multiple email identities, while only having to retrieve authentication tokens from a single account, thus providing a form of multiple-identity single sign-on.

Selective delegation provides finer-grained delegation. In this model, only token messages that meet a certain criteria are forwarded. For example, only messages with tokens from site Y are forwarded to delegate X. This method allows the

user to specify whether or not attempts to change the primary/secondary email address registered at a provider should be forwarded, thus preventing delegates from assuming complete control, unless explicitly allowed by the delegator. When selectively delegating authority it is important to intelligently forward tokens to the authorized user who actually initiated the request. This avoids superfluous messages to all involved.

A valid email address [8] allows for the addition of a sequence of words, called a *phrase*, before the actual email address. This has traditionally been used to create a pretty-printed form of a real name that corresponds to the email address, e.g., "William Henry Gates III" <bgates@microsoft .com>. By modifying the contents of this phrase, the intended target of the authentication mechanism is easily identifiable. For example, suppose Linus Torvalds has been authorized by Bill Gates to access a particular web site. When Linus supplies the value:

"Delegate to: torvalds@osdl.org"<bgates@microsoft.com>

as his email address, Bill's email filter is able to trivially determine whether a particular authentication message is meant for him or for Linus.

This method also facilitates the creation of groups of users. Although this approach has the potential to remove the anonymity of group members, it also avoids everyone in the group from getting the authentication messages from everyone else in the group. It must be noted that this solution also allows web sites to be aware that delegation is occurring and possibly act on it. This may or may not be desirable to the user.

One-time delegation is a subset of selective delegation. The delegator begins the authentication process and collects the required authentication tokens, but instead of submitting them to the web site, the delegator supplies them to the delegate. Due to the short lifespan of the tokens, this process is very time-sensitive.

C. Client-side Auditing

Since current password authentication only involves the web site and the entity attempting to authenticate, which may or may not be the user, there is no means, without the support of the web site, to provide audit information to the user concerning authentication attempts and failures.

In SAW, it is possible for the user to audit the authentication process without any support from the web site because any such attempt must necessarily pass through the user's email account. This auditing capability remains available, even when authority is delegated using client-based delegation. By saving a copy of the messages that are forwarded, a client-side audit trail is established. Alternatively, with minimal modification to the site, an audit message (e.g., an authentication message with the AuthToken removed) sent to a user's secondary email address also creates a client-side audit trail. If a web site has built-in delegation capabilities, the client-side auditing of SAW may not work unless the site notifies the user each time a delegate accesses the system.

D. Active Impersonation Countermeasures and Privacy-Enhancing Features

This section describes some optional mechanisms that prevent active impersonations by email providers and add *confidentiality* and *anonymity* to the email messages. These mechanisms are not mandatory because we believe that the increased overhead and complexity they add is too costly to address risks that the large number of EBPR-enabled sites have already demonstrated are manageable.

First, it must be noted that in order to avoid detection an active attacker must intercept and remove email messages destined for the user. Otherwise, a client-side auditing mechanism would detect this breach. This is arguably more difficult than simply passively observing this link.

To prevent active impersonations, SAW can concurrently involve multiple email accounts in a single authentication. The secret splitting scheme used to create the authentication tokens is easily extended to n email accounts as follows:

$$AuthToken_{user} = AuthToken_{email_1} \oplus \cdots \oplus AuthToken_{email_n} \oplus AuthToken_{complete}$$

If each $\text{AuthToken}_{email_i}$ is sent to a different email account, active impersonations require the eavesdropping of, or the collusion of, all the providers of these accounts. Choosing email providers in unrelated domains greatly decreases the possibility of this occurring.

Also, by using a threshold secret sharing scheme [5], it is possible to split the secret such that the user only has to retrieve m of n messages to successfully authenticate. This has the potential to improve latency when one or more email providers are busy or unavailable. Nonetheless, splitting AuthToken_{email} between multiple accounts diminishes the single sign-on potential of SAW.

Note that the location where the actual passive eavesdropping occurs affects the payoff to an attacker. If an attacker is observing the incoming traffic to a particular email provider, she is easily thwarted by using the multiple email account scheme described above. On the other hand, if the attacker is listening to all outgoing email traffic of a particular web site the multiple email account scheme would not be effective, but the attacker can only impersonate the user at that particular site, not at the other sites that can be accessed by the user.

To optionally provide confidentiality and anonymity to the delivery of AuthToken_{email}, the message containing it is encrypted and delivered via an anonymous remailer. Users decrypt this message using the decryption key that accompanies AuthToken_{user}. By encrypting this message and hiding its origin, users prevent their email providers from gleaning information about an authentication at the expense of increased latency.

VI. Overcoming Email Obstacles

Latency: Although typical email message delivery is nearly instantaneous, SAW is not immune to high traffic conditions and other factors (e.g., virus/spam filters, archiving) that delay message delivery. Some corporate email systems support the

designation of high priority messages to reduce latency. When such support is unavailable, using instant messaging in lieu of email is an effective method to avoid the potentially high latency of email. While SAW may not suitable for every email system, our initial performance results demonstrate feasibility of SAW as a primary means of authentications, even on a highly loaded email system like Gmail.

Privacy: Section V-D enumerates the privacy enhancing features available to SAW. These features, as well as the protections provided by the protocol itself, allow SAW to detect and prevent threats to a user's privacy, even when the link between the web site and the email provider has no cryptographic protection.

Convenience: While it is difficult to compete with something as pervasive and embedded as password-based authentication, SAW provides significant benefits to justify the switch and is designed to minimize the cost of such a transition. Most notably, no modification or changes are required to email providers. No specialized client-side software is required, although the convenience and benefits of this system are greatly enhanced through client-side automation. The only entity that requires modification is the web site.

Web sites already have the necessary hooks to both the local applications and the email system to implement SAW. Transitioning to SAW requires a repurposing of services that are already in use. The most significant modification to this module is the addition of the ability to create authentication tokens, which is purposely straightforward and computationally light-weight.

Pervasive user training on email-based authentication began many years ago with the adoption of EBPR. Users are already well aware of the utility and convenience their email accounts hold for authentication. SAW exploits this foundation to extend and improve this form of authentication.

Reliance on a third party: The reliance of SAW on email providers is mitigated by the distributed and decentralized nature of email providers. The use of a secondary email address alleviates the problems caused by temporary lapses in availability of a user's primary email provider. Although an email provider could selectively drop or refuse authentication messages, this issue is between the email provider and the user and is outside the scope of SAW.

VII. Related Work

PKI-based systems overcome many of the problems associated with passwords. By specifying trusted CAs, and using certificates issued by those CAs for client authentication, systems like client certificates in TLS have the potential to overcome the weaknesses of password-based systems. Unfortunately the adoption of PKI-based systems by businesses and the general public has been slow. PKI systems have burdensome configuration and usage requirements [9]. High costs, the requisite user and administrator training in a complex technology, and the difficulties of trusted roots and cross-certification also makes PKI hard to adopt [10]. Ellison and Schneier [11] further illustrate the risks of PKI and "trusted" CAs. Overall, PKI-based systems are too heavy-weight for most web sites and their users.

Centralized single sign-on is seen by many as the panacea to the rising password woes. Systems like Liberty [12] and Shibboleth [13] use specialized identity servers that provide authentication and the ability to assert additional attributes about their users. Though promising for future applications, the legal and business relationships required to implement these systems prohibit their widespread adoption. At this stage these systems are too heavy weight to replace passwords as the preferred authentication system. SAW does not require any legal/business agreements between the identity provider and the web site. Such agreements may improve the benefits of SAW, but are not necessary to deploy and use this system.

There are a variety of URL-based authentication systems. These systems, such as OpenID [14], Light-Weight Identity (LID) [15], and Simple eXtensible Identity Protocol (SXIP) [16], uniquely identify a user using a URL. Based on this URL, web sites communicate with a specialized identity provider to verify user identity. Although these systems provide a decentralized authentication mechanism, they require the creation of specialized identity providers and new identifiers (i.e., URLs) that are not as intuitive, recognizable, or as widely used as email addresses and other personal messaging identifiers. SAW is a simpler alternative to these systems when the authentication does not require any attribute information exchange. Also, SAW can serve as the authentication method for the identity provider, avoiding the need for yet another password.

Wordpress [6], a popular web log platform, has a third-party plug-in, Comment Authorization [17], that is designed to reduce comment spam by requiring self-moderation of comments. Once a comment is posted, an authorization link is sent to the author's purported email address. By clicking the link, the owner of the email address authorizes the posting of the comment. This method of self-authorization presents a novel idea for distributed authorization, but it is not a general purpose authentication mechanism like SAW.

Our primary research focus for the past decade has been trust negotiation [18], an approach to authentication in open systems that relies on third party attribute assertions contained in digital credentials. Trust negotiation has a significant deployment hurdle. Users typically do not possess digital credentials because most web sites do not accept them; most web sites do not accept certificates because users do not possess them. It is unclear when, or if, this chicken and egg problem will be overcome. SAW grew out of our search for simpler approaches with the potential for an immediate impact on authentication challenges in open systems.

VIII. Conclusions

SAW is a simple concept. It builds on EBPR, which has already proved its utility for authenticating users, and improves it by thwarting passive attacks and significantly raising the bar for active attacks. These enhancements make SAW a viable alternative for passwords at a significant number of

web sites. SAW has the potential to thrive because it does not require universal acceptance, modification of email providers, nor significant changes to existing web site infrastructure.

SAW is an important step towards simplifying authentication and making it more convenient. It relieves both web sites and users from having to establish and manage passwords by off-loading user authentication to email providers. Although SAW does not eliminate passwords completely (users still have to authenticate to their email providers), it should greatly reduce the number of passwords users need to remember. In addition, this system provides single sign-on to all SAW-enabled sites, exploits client-side automation to speed up the login process, and reduces the attack surface for phishing and social engineering attacks.

SAW addresses all the obstacles enumerated in Section II-A that would impede the adoption and utility of this new web site login approach. It also provides several advanced features not usually available to password-based systems. These include: 1) Leveraging email addresses and proof-of-address-ownership to facilitate sharing and collaboration; 2) Using email forwarding rules to allow intuitive delegation and revocation of authority; and 3) Exploiting the fact that all authentication attempts pass through a user's email account to provide client-side auditing.

SAW is a simple concept, but therein lies its strength. It is easy to understand and to implement. Its benefits are significant. The risks of its use are clear and have already proven to be manageable.

A. Future Work

As mentioned earlier, SAW provides personal messaging-based authentication and easily translates from email to other mediums, e.g., instant and text messaging. We are currently exploring other personal messaging systems and the possibilities of a hybrid combination of these mediums.

For example, a cell phone with email or text-messaging capabilities has the potential to mitigate the risks of using SAW from an untrusted machine, such as in a cyber café or public library. By retrieving AuthToken_{email} from the phone, or having the email provider forward it to the phone, the untrusted device will never be able to capture the email account password. To be effective, a means to conveniently transfer the token from the phone to the computer must be devised, e.g., transforming the token into a more human-friendly form or using a limited-range wireless protocol.

SAW primes the pump for attribute-based authentication because some email providers (e.g., universities and businesses) are authoritative sources for certain user attributes. In fact, the mere possession of an account at a specific provider is a desirable attribute for specifying access control. Email aliases are one technique to assert these attributes without modifying the email provider. For example, Alice has the account alice@cs.someschool.edu. This email address may be used to learn that Alice is a computer science student or faculty member at someschool. By creating the alias: alice@phd.grad.cs.someschool.edu, more attributes are added.

Alternatively, by replacing the delegation string described in Section V-B with an attribute(s) request, the email provider could be modified to deliver messages only if the recipient has specific attributes. In either of these approaches, the trustworthiness of the attributes being asserted depends on the relationship between the email provider and the web site.

Since a variety of applications, not just web site logins, benefit from SAW, it would be convenient for users to have a centralized, client-side location to retrieve and distribute AuthToken_{email}. A local email client is an excellent location to host such a service. Through an extension or plug-in this email client (e.g., Mozilla Thunderbird and Outlook) would provide a single, intuitive location to manage local access to email accounts. It also provides a location to enforce fine-grained permissions on the programs that are allowed to access the authentication information sent from a particular web site.

A variety of interesting and exciting possibilities become a reality when email providers take an active role in the authentication process. Specifically, we are interested in making SAW feasible even when users cannot directly contact their email providers, e.g., logins for wireless internet access.

We are also currently preparing a large scale user study to evaluate the usability of SAW. This study will focus on authenticated blog access and file sharing.

ACKNOWLEDGMENTS. This research was supported by funding from the National Science Foundation under grant no. CCR-0325951, prime cooperative agreement no. IIS-0331707, and The Regents of the University of California. We thank Andrew Harding for his work on the browser extensions, Daniel Wille for his work on the WordPress plug-in, and Reed Abbott for his work on the Apache authentication module.

REFERENCES

[1] "Password Safe," http://schneier.com/passsafe.html.
[2] S. L. Garfinkel, "Email-based Identification and Authentication: An Alternative to PKI?" *IEEE Security & Privacy*, pp. 20 – 26, 2003.
[3] T. Wu, "The Secure Remote Password Protocol," in *Network and Distributed System Security Symposium*, 1998, pp. 97–111.
[4] M. Steiner, G. Tsudik, and M. Waidner, "Refinement and extension of encrypted key exchange," *SIGOPS Oper. Syst. Rev.*, pp. 22–30, 1995.
[5] B. Schneier, *Applied Cryptography, 2nd Edition*. New York, NY, pages 70-71: John Wiley & Sons, Inc., 1996.
[6] "Wordpress," http://wordpress.org/.
[7] P. Saint-Andre, "RFC 3921: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence," , United States, 2004.
[8] P. Resnick, "RFC 2822: Internet Message Format," , United States, 2001.
[9] P. Gutmann, "PKI: It's not dead, just resting," *IEEE Computer*, vol. 35, no. 8, pp. 41–49, August 2002.
[10] "US General Accounting Office, Advances and Remaining Challenges to Adoption of Public Key Infrastructure Technology, GAO-01-277, 2001; http://www.gao.gov/new.items/d01277.pdf."
[11] C. Ellison and B. Schneier, "Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure," *Computer Security Journal*, vol. 16, no. 1, pp. 1–7, 2000.
[12] "Liberty Alliance Project," http://projectliberty.org/.
[13] "Shibboleth," http://shibboleth.internet2.edu/.
[14] "OpenID," http://openid.net/.
[15] "Light-Weight Identity (LID)," http://lid.netmesh.org/.
[16] "SXIP 2.0 Protocol Specification," http://sxip.net/.
[17] "Comment Authorization," http://wp-plugins.net/plugin/commentauth/.
[18] W. H. Winsborough, K. E. Seamons, and V. E. Jones, "Automated Trust Negotiation," *DARPA Information Survivability Conference and Exposition*, Jan. 2000.