Adaptive Semi-Private Email Aliases

Beng Heng Ng, Alexander Crowell, Atul Prakash
Department of Computer Science and Engineering
University of Michigan
{bengheng, crowella, aprakash}@eecs.umich.edu

ABSTRACT

Email address leakages are the cause of several security problems including spam and privacy loss. With current email addresses, once the address leaks without its owner's consent, it becomes effectively compromised, creating a struggle for the user to keep the address out of the hands of new spammers. To compound the problem, some websites require addresses belonging to a certain domain (e.g., `<university>.edu`) as a partial proof of the user's affiliation with an organization. This leaves the user not much choice except to have faith that the address will not be misused.

To address the problem, this paper improves on the prior work on disposable email addresses by proposing a mechanism called *semi-private aliases*. Semi-private aliases make two contributions. First, they have a lifecycle model that permits gradual, selective controls on the use of the alias by senders without requiring any special infrastructure on the part of senders or receivers. Second, semi-private aliases can be easily used to validate a user belonging to a certain organization (e.g., university or company) and reveal only selected attributes to a service while hiding the real identity. The second aspect recently proved useful in allowing students in one of our freshmen courses to register easily and safely at Piazza.com, a discussion forum for courses, that, by default, requires students to provide a university email address, but has privacy policies that differ from a university's.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*; H.4.3 [**Information Systems Applications**]: Communication Applications—*Electronic Mail*

General Terms

Design, Security

Keywords

Semi-Private, Alias, Email Aliases, Lifecycle, Affiliation Validation, Disposable Email Addresses, Email Address Leakages, Spam, Unsolicited Email

1. INTRODUCTION

Since its inception in the 1970s, electronic mail, or email, has come to largely replace snail mail for a variety of reasons, including its low cost, user convenience, ease of use, and high delivery efficiency. But for these very same reasons the problem of unsolicited bulk email messages, commonly referred to as spam, has grown along with email since the 1990s, spurring huge research efforts for finding tools to combat spam. Although the state-of-the-art in spam detection has been successful at detecting most obvious cases of spam, false positives and false negatives are still not entirely uncommon.

One primary cause of the spam problem is the way in which email addresses are typically used. To provide a fixed address at which they can be reached, most users treat their email IDs as permanent and only abandon them in rare circumstances. As a result, once a user's email address leaks to spammers, it is nearly impossible to entirely prevent them from sending messages to the user's inbox. Users usually have no recourse if they wish to retract the release of an email address to a certain party. And although one may take extreme care to prevent one's email address from falling in to the wrong hands, because anyone with the address could then leak it to a third party, either intentionally or unintentionally, such an effort can easily be futile. The situation becomes further complicated if we consider that a user may later change their mind about whether they should have given their email address to a certain party.

To compound the problem, some services require corporate email addresses as part of the proof of a user's affiliation with the organization before deeming the user as eligible for certain services or discounts. Early examples included Facebook, which required university affiliations when it started. More recently, Piazza.com is used by many universities to host threaded forums between students and professors and, by default, it requires users to sign in with an email ID that validates them to their university. This creates a potential dilemma for professors as to whether it is proper to require students to sign up for an external service with a different privacy policy on protection of students' email IDs than their university's. Many companies offering corporate discounts on their services or products, for example, Sprint and AT&T in the U.S., also require customers to provide their corporate email ID to receive discounts on their monthly bills. While alternative proof methods may be allowed, these are usually troublesome. This also creates concerns: a corporate address is potentially being used to receive non-work email, making it more susceptible to marketing use.

This paper proposes a mechanism called *semi-private aliases*, a novel solution that attempts to blend the user control provided by disposable email addresses with the flexible nature of ubiquitous permanent email addresses to provide an email aliasing mechanism that can limit misuse without being overly restrictive to either the address owners or their trusted correspondents. Semi-private aliases are email addresses that can be attached seamlessly to a user's regular inbox, and serve as aliases for that inbox that can be distributed in the same fashion as Disposable Email Addresses (DEAs) [1, 2]. These aliases make two significant contributions:

1. *Concept of alias lifecycle:* A *lifecycle*, in the form of a state machine, allows the user to adaptively add restrictions to an alias as the effects of an address leak begin to show themselves. Starting out as *unrestricted* and open to all incoming mail, an alias can be marked *partly restricted* when unsolicited email begins to be received, resulting in no added restrictions for those who have corresponded with the alias before the identified compromise. New senders sending to a partly restricted alias receive a CAPTCHA challenge, after which the user is prompted to accept or reject that sender's correspondence. Once an alias has reached the point where the user does not expect any new contacts on it, they may mark it as *fully restricted*, at which time only those contacts on the alias' whitelist are permitted to send. Finally, a user might choose to *disable* an alias if it eventually falls entirely out of use.

2. *Privacy-protecting affiliation validation:* When deployed by an organization, the service can be used by individuals in the organization to validate their organizational affiliation (and optionally a selection of other information such as their name or role in the organization) to external service providers without the risk of exposing the real corporate email ID information to the providers. Instead, service providers are provided with an email alias.

Some challenges in designing our system for semi-private aliases, which we call SEAL, are how to make it work with existing email infrastructure and services as well as how to avoid requiring significant authentication steps (e.g. CAPTCHAs) along communication paths. SEAL achieves these goals, using encrypted SMTP supported by existing infrastructure, including Gmail and software clients such as Mozilla Thunderbird, for sending messages; for authentication of senders to aliases, we make use of alias lifecycles to identify when senders are likely to be potential spammers and execute authentication steps only in these cases.

We note that SEAL primarily aims to give users selective control over their privacy rather than provide complete anonymity over the web. We assume that the SEAL infrastructure is trusted, but we attempt to design it so that the theft of information in the database maintained by SEAL for its functioning is of limited use to spammers.

The paper is structured as follows. We first discuss related work in Section 2. Then we present SEAL from an end-user's perspective in Section 3. We also discuss the design and describe our prototype in Section 4. Next, we evaluate the effectiveness of the system in restricting aliases and tracing alias leakages, and the deployment of our system in a real world scenario in Section 5. Finally, we discuss potential limitations and defenses against potential attacks on the SEAL design in Section 6 before concluding in Section 7.

2. RELATED WORK

Variants of disposable email aliases are supported by several systems. We divide current DEA solutions into two groups, characterizing them as either *incomplete* or *overly restrictive*. The first category of DEA systems are specialized services that allow users to create DEAs but do not provide full email services. They can be sub-categorized into receive-only systems that do not allow a user to reply to emails, and temporary systems that only allow users to access their inbox for a limited amount of time [3]. Others, e.g., Mailinator [4, 5], provide a single shared inbox for all users, and so there is no notion of privacy; anyone with the email ID string can access the email to that email ID.

In the second category, the DEA systems are overly restrictive, either only permitting the complete removal of an address to prevent spam or requiring that every correspondent solve a CAPTCHA. One example of such work is Inexpensive Email Addresses (IEA)[6]. IEA cryptographically generates exclusive email addresses for each sender that must be verified by CAPTCHA. This greatly limits the system's practicality, making it difficult to use with automated systems like mailing lists, newsletters, and password recovery services. For normal users, it requires them to go through an extra step of solving a CAPTCHA before being able to send an email. Yahoo Mail's aliases require removal of an alias to prevent spam, once traditional filters break down.

Ioannidis proposed the concept of a Single-Purpose Address (SPA) [7], where an SPA has cryptographic properties and encodes security policies that can be enforced by receivers into the email address itself. When a receiver creates an SPA, an expiration date is supplied that determines its lifetime. This encoding of policy into SPAs severely limits their usefulness; an encoded policy can never be changed during the life of the email address, so a single compromise means the owner must live with spam until the address expires or switch to a new key, thereby invalidating all of his existing SPAs. The unlikely event of a server compromise also poses a much greater problem for SPAs, since the leak of a key requires the owner to invalidate all of their SPAs and start from scratch. SEAL avoids these limitations by keeping state on the server, allowing the user to flexibly respond to address leaks by restricting specific aliases. A side benefit of this is that in the event of a temporary compromise of the server itself, once control is regained the user can restrict all of their preexisting aliases, avoiding a temporary complete loss of service due to the need to create and distribute entirely new email addresses.

The Tagged Message Delivery Agent (TMDA) is a challenge/response system that aims to mitigate spam [8]. One feature of TMDA is tagged addresses that can contain date information used in a similar manner to SPA for determining the expiration of the addresses. Similar to SPA, the expiration date has to be determined at the point of creation. Again, the tagged address may be leaked before it expires.

The free online classified advertising site Craigslist generates a random anonymous email address for the user when a posting is made. While this conceals the user's real address from email harvesters that scrape websites, Craigslist is often attacked by spammers who post fake advertisements. An unknowing user who replies to the fake email reveals his real address. With SEAL, a user can simply use a semi-private alias when creating or responding to an ad on Craigslist, giving them the ability to block spam sent to that address at any point.

OpenID [9] permits users to sign up for external services using an existing email ID, such as their Facebook ID or Google ID, while providing some privacy controls. SEAL accomplishes a similar goal but without requiring the service provider to use a specific authentication infrastructure and having full control over the information that is disclosed with the email alias.

Spam filtering has been well-studied [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24], and is complementary to our approach. While SEAL is not a spam filter, the life-cycle management controls provide an additional layer of spam control when traditional spam filtering fails, without requiring an alias to be com-

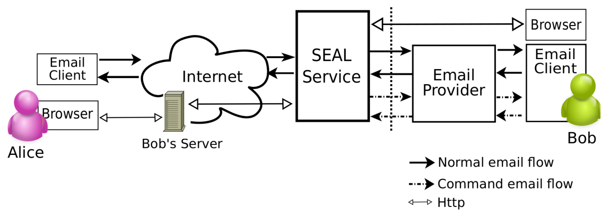


Figure 1: Overview of the Seal Service

pletely disabled.

3. USER'S PERSPECTIVE

Senders correspond with users of SEAL using semi-private email aliases. An example of such an alias would be `bob.89dtzx3r@sealserver`, where `bob` is the *alias name* and `89dtzx3r` is the *randomization string*. An alias is formed by joining the alias name and the randomization string with a delimiting character in between. The alias name is specified by the user while the randomization string is a randomly generated string, created by SEAL.

Figure 1 shows an overview of the email interactions between a SEAL user and a sender who wishes to correspond with that user. A user of the SEAL service sends mail through a SEAL server that processes the mail and forwards it to the recipient. A person sending to a SEAL user addresses their mail to the user's semi-private alias, and the SEAL service performs any necessary restrictions, after which the mail can be forwarded to the user at their normal inbox. Users can also manage their aliases directly over a provided web interface.

To become a user of SEAL, one creates an account with an email provider and configures the account to relay emails through SEAL. The system could also potentially play the role of an email provider. However, segregating the roles has two practical advantages.

Reduced attack surface: Leveraging existing email providers allows SEAL to obviate the need to provide message storage. This reduces the attack surface of SEAL and also insulates the user's emails from theft or corruption in the event of an attack.

User familiarity: Most users are already familiar with the user interfaces of their current email providers. Many email providers also provide other useful features. Using the services of existing email providers for SEAL eliminates the need for users to learn a new interface and allows them to continue using their favorite features.

To achieve this, we require the mail provider to support sending mails over authenticated SMTP, which is supported by some email providers, including Gmail, as well as most modern software-based email clients such as Mozilla Thunderbird. This is necessary to prevent masquerading attacks on SEAL.

In addition to normal emails, the user can also send command emails to two Service Addresses that allow the user to provide instructions to SEAL. Users can also receive feedback on the commands from these Service Addresses. Table 1 lists the commands supported. When an email is received at a Service Address, only the Subject line is parsed for commands. Table 2 summarizes whether a sender is allowed to send email to a semi-private alias for each of the different alias states. Figure 8 in the Appendix shows the state transitions of an alias.

Table 1: Commands used by SEAL. All commands are specified in the email Subject line. The contents of the message bodies are ignored.

Command	Service Account	Subject Line
Request alias	getalias@sealserver	⟨alias name⟩
Partly restrict alias	service@sealserver	restrict ⟨alias⟩
Fully restrict alias	service@sealserver	restrict full ⟨alias⟩
Trust sender	service@sealserver	trust ⟨email address⟩
Distrust sender	service@sealserver	distrust ⟨email address⟩

Table 2: Capability matrix between sender status (columns) and alias states (rows). A '✓' denotes that the sender is allowed to send to the alias, while a '✗' denotes the contrary. `CAPTCHA` denotes that the sender will be arbitrated to be trusted or not by solving a CAPTCHA challenge and receiving explicit permission from the user.

	Distrusted	Unknown	Trusted
Unrestricted	✓	✓	✓
Partly Restricted	✗	`CAPTCHA`	✓
Fully Restricted	✗	✗	✓
Disabled	✗	✗	✗

3.1 Lifecycle of a Semi-Private Alias

After creating a user account, the user can request an alias name that is not in use by other users. Using the alias name, the user can request aliases for distribution to contacts. Figure 2 shows the lifecycles for three aliases. At t_1, the user requests a new alias for the alias name `bob`. We discuss the different methods for requesting a new alias in Section 3.3. The system returns the new alias `bob.rzkyt7y4` which can be distributed to the user's contacts. The user corresponds with the contacts on `bob.rzkyt7y4` until he observes that it has been leaked at t_2 and informs SEAL via a command email. SEAL then marks all senders prior to t_2 as *trusted*, marks the alias as partly restricted, and generates the successor alias `bob.u1pvwf47`.

At this point, it may be possible that spammers prior to t_2 are erroneously marked as trusted. However, this is reversible. The user can refine which senders should be trusted. Another possible approach may be to let the user decide the earliest time when the first spam to the alias is found and to mark all senders prior to that time as trusted. However, if the user makes a mistake in finding the first spam, mail from legitimate senders may be blocked, especially for automated systems like mailing lists. Therefore, we decided to

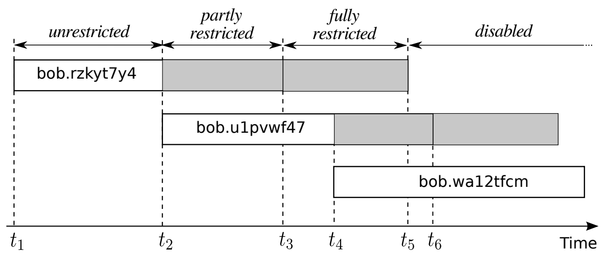


Figure 2: Lifecycle scenarios of three aliases. The unshaded part of a bar shows the alias is *unrestricted* while the shaded part shows that it has been leaked and becomes *restricted*.

take the more conservative approach.

Between t_2 and t_3, SEAL checks that the senders of email addressed to the leaked `bob.rzkyt7y4` are trusted before relaying their mail to the user. This also indicates that the sender has not updated his address book to send to the new unrestricted alias `bob.u1pvwf47` and so upon the user's reply, SEAL sends the original sender a reminder of the change. Email from untrusted senders is dropped. If the sender is neither trusted nor untrusted, we drop the mail and send a CAPTCHA to the sender. If the sender solves the CAPTCHA, a command email is sent to the user seeking permission to trust the sender. When the user agrees, the sender is added to the group of trusted senders and notified. Requiring the sender to solve a CAPTCHA first prevents the user from being overrun with requests, narrowing them down to requests only for senders who are likely to be human. Requiring the user's approval to add the sender ensures that consent is explicitly given.

At t_3, the first alias, `bob.rzkyt7y4` is changed to the fully restricted state. In this state, only trusted senders can successfully send emails to the user. No new sender can become trusted as no CAPTCHA will be issued. At time t_4, `bob.u1pvwf47` is found to be leaked. The new successor `bob.wa12tfcm` is created. Trusted senders still sending to `bob.rzkyt7y4` will receive the notification to update the address of the user to the newest successor, which in this case is `bob.wa12tfcm`.

At time t_5, the original alias is disabled and no emails will be delivered through it to the user. However, we do not expect this to be a common operation since aliases in fully restricted mode allow the user to maintain communication with already trusted users, and also due to the likely overhead of distributing a new email address to a user's correspondents. Hence, so long as the user does not wish to notify anyone else of the locked-down alias it can continue to be used without the generation of a replacement. More importantly, no new senders sending through the alias would be marked as trusted, thus essentially denying them from emailing the user.

3.2 Aliases as Proof of Affiliation

As mentioned in the introduction, semi-private aliases can also be used as a means of offering an *affiliation validation* service, which provides proof of a user's affiliation with some organization in order for them to gain access to certain services or discounts. Trivially, our implementation could be extended further, with the organization providing an additional service that allows its members to attach a supplementary information profile to each alias and host that profile on a directory service as more detailed proof of the identity associated with a certain alias address. At this time, with our current real-world deployment that gives students access to Piazza.com forums, we have only needed to provide basic affiliation validation.

3.3 Requesting an Alias

The user may need to distribute aliases under a multitude of scenarios. We broadly categorize them as *online* and *offline*. By online, we mean that the user needs to distribute an alias while she has network access to SEAL's server. In contrast, in an offline scenario, the user is not able to interact with SEAL over the network. However, we assume that there are opportunities for the user to access SEAL at some point in time prior to needing an alias. We subcategorize online scenarios into alias requests and retrievals. An alias request creates a new alias that the user can distribute to a new contact. An alias retrieval refers to scenarios where the user wishes to retrieve a previously requested alias associated with a website URL. To minimize the learning curve, it is important that minimal effort be required from the user when requesting new aliases under the different scenarios. We now give an overview of the most likely scenarios and describe briefly four mechanisms provided by SEAL to request aliases under different scenarios.

3.3.1 Request via Command Emails

In an online scenario, a user who has access to an email client can send command emails to the service address `getalias@seal-server` to request a new alias. This is the catchall mechanism since we can assume that users will normally have access to some email client. The server responds with an email containing an alias that the user can distribute to contacts. SEAL's response would be stored in the user's inbox. We also allow the user to specify a *hint* as a reminder to the context under which the alias is generated. Figure 9 in the Appendix shows a request example on the left and the server's response on the right.

3.3.2 Request via Browser Extensions

Another common online scenario requires the user to request an alias which may be subsequently needed for identification purposes. Specific examples of such a scenario include accessing the user's account information for a website and posting on forums. To cater to such situations, SEAL provides a browser extension that uses magic sequences for alias request and retrieval.

To detect the magic sequences, we use the same approach as PwdHash [25]. In that work, a browser extension transparently generates a unique password for the user. The other functionalities of the two extensions differ. We developed a Firefox extension that operates in two modes, *request* and *retrieval*, triggered by two magic sequences. For each browser session, when a magic sequence is detected, the extension authenticates the user with our system via their credentials. Once authenticated, a session key is generated and stored by the extension for the current session. Request mode automatically fetches a new alias from our server and is triggered by typing the magic sequence "`##[alias]#[hint]#`". A salted hash of the site's domain is stored. Retrieval mode is triggered by the magic sequence "`##$`" and is used when the user logs in to a previously registered site that requires an email address for authentication. The salted hash of the domain is used for looking up the previously created alias.

3.3.3 Request for Offline Distribution

The most challenging scenarios occur when a user is offline. For example, the user could be filling out a paper form at some place lacking an Internet connection. Though we have not implemented this, we envisage an SMS service that replies with a new semi-private alias whenever the user makes a request. In addition, a mobile application that caches several aliases while it has network access and dispenses them as needed could be used.

An even more challenging scenario is posting email IDs on web pages, printed documents, or on business cards. Since such IDs are widely disseminated, they are likely to generate both spam and legitimate use very quickly, even if the IDs are semi-private aliases. We discuss a potential solution to the problem in Section 6.

4. ARCHITECTURE

SEAL's architecture is illustrated in Figure 3. The three main components in SEAL's core architecture are the `Dispatcher`, `Email Processor`, and `Command Processor`. The `Dispatcher` receives email over SMTP and passes them to the appropriate modules. If the email is a normal email, it is dispatched to the `Email Processor`. Otherwise, a command email is sent to the `Command Processor`. Other than using emails, it is also possible to interact with the `Command Processor` over HTTP/S. We

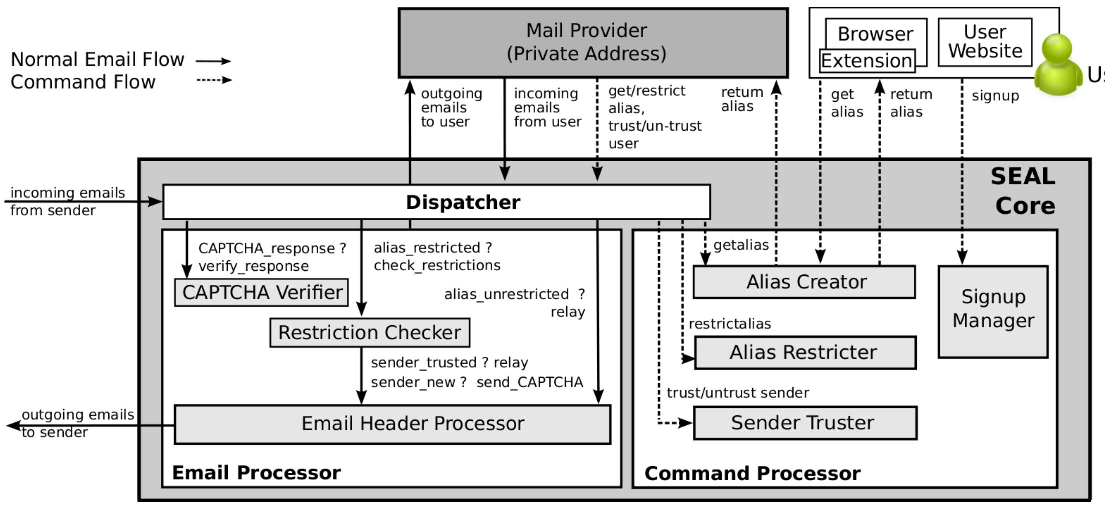


Figure 3: SEAL architecture.

discuss the components with reference to their functionalities.

Figure 4 shows a simplified version of our database. Each user has a salt that is used for hashing sensitive information, such as the sender's email addresses. This is to limit potential information loss in the event that SEAL is compromised.

4.1 Account Creation

The user creates a SEAL account by visiting SEAL's signup page and specifying the username, password, and relay address. The `Signup Manager` records this information to the database. The username and password are used for SMTP authentication by the user's mail provider when sending email through SEAL, with the username being converted to a sender address taking the form `<username>@sealserver`. It is worth noting that this email address constructed from the username could be revealed to arbitrary recipients via reply messages or other means, but such a leak would not compromise the user's account since email sent to that address is simply dropped. All incoming messages to aliases and replies to command emails are sent to the relay address. By storing only the basic necessary information for SEAL's proper functioning, we aim to minimize the risks of theft of sensitive user data should our server ever become compromised. While our system works with any existing email account whose provider supports sending email as a user of another SMTP server, ideally, a new account should be created so as to start from a clean slate since the existing address might have already been leaked.

4.2 Alias Request

Requests for new aliases are sent to `Alias Creator`. This could be done either using a command email or an HTTP GET Request. The `Alias Creator` takes an alias name and an optional hint as inputs. If the alias name has not been taken by another user, `Alias Creator` creates a randomization string of length eight. We allow 32 possible alphanumeric case-insensitive characters (excluding '0', 'o', 'i', and 'l' to avoid potential user confusion) in the randomization string, providing a base entropy of 2^{40} bits for each alias address. This randomization helps to make it difficult for a spammer to correctly distinguish valid aliases from invalid ones. Implicitly, since the maximum allowable length of an email ID is 64 characters and a delimiter is used, this restricts the alias name to a maximum of 55 characters.

We note, however, that since the randomization string exists primarily to help thwart guessing attacks, there is a tradeoff that can be made by those deploying a SEAL server between the reduced guessability provided by longer randomization strings and the improved readability and greater potential length of alias names provided by short randomization strings. We chose a length of eight characters as an heuristic compromise between the two, but the length could easily be made shorter or the randomization even eliminated entirely if it is not felt to be of particular importance. One possible design could be to allow the user the liberty to generate her own random string. However, an adversary could then guess existing aliases trivially.

The optional hint replaces the user's name in the `To` header and can be used to remind the user of the context for which the alias is intended. Figure 10 shows an example of a hint "work". To prevent the original sender from observing the hint, it is removed in the reply mail to the sender.

4.3 Managing the Alias Lifecycle

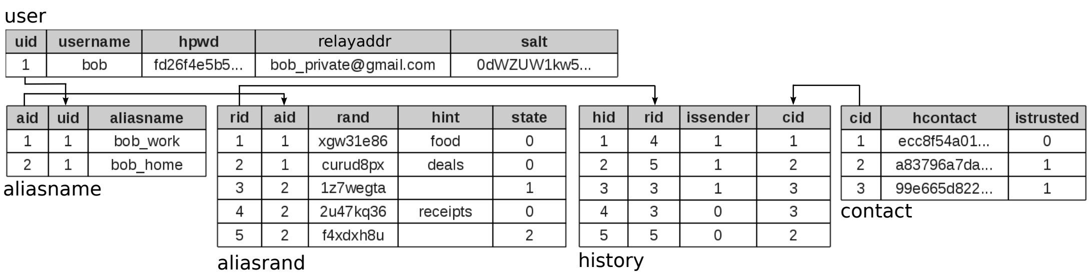
An alias' lifecycle begins when the user makes a request. `Alias Creator` then creates an entry in the database.

When a new email is received for an alias from a non-user, the `Dispatcher` checks the state of the alias. If it is unrestricted, `Email Header Processor` (`EHP`) replaces the `To` header with the user's relay address and appends the alias to the `Reply-To` header before sending it out. This causes the user to send their reply to the alias, which will result in SEAL processing the reply mail to appear as if it had been sent by that alias.

On the other hand, if the alias is restricted, the email is dispatched to the `Restriction Checker`, which then checks if the sender is trusted. If it is, the email is relayed to the email provider via `EHP`. If the sender has not yet been encountered by the recipient alias, the `Restriction Checker` tells `EHP` to generate a CAPTCHA response for the sender. Otherwise, the sender is untrusted and the email is dropped.

If an email arrives from a user, it is dispatched directly to `EHP`, which will replace the `From` header with the alias specified in the `To` header, so long as it is owned by that user.

If `Dispatcher` detects that the incoming email is a response to a CAPTCHA challenge, it forwards the email to the `CAPTCHA Verifier`, which will validate the response and send a system message to the user to confirm the sender as trusted. While waiting for user validation, the sender will be treated as untrusted.



user

uid	username	hpwd	relayaddr	salt
1	bob	fd26f4e5b5...	bob_private@gmail.com	0dWZUW1kw5...

aliasname

aid	uid	aliasname
1	1	bob_work
2	1	bob_home

aliasrand

rid	aid	rand	hint	state
1	1	xgw31e86	food	0
2	1	curud8px	deals	0
3	2	1z7wegta		1
4	2	2u47kq36	receipts	0
5	2	f4xdxh8u		2

history

hid	rid	issender	cid
1	4	1	1
2	5	1	2
3	3	1	3
4	3	0	3
5	5	0	2

contact

cid	hcontact	istrusted
1	ecc8f54a01...	0
2	a83796a7da...	1
3	99e665d822...	1

Figure 4: Simplified SEAL database. In table `aliasrand`, the states 0, 1, and 2 mean unrestricted, partly restricted and fully restricted respectively.

The user may mark a particular alias as partly restricted or fully restricted. The user does this by sending a command email to `service@sealserver`, which will be dispatched to the `Alias Restricter`. Similarly, the user may mark a sender as trusted or untrusted. The command is dispatched to `Sender Truster`. Note that the restriction level for an alias is monotonically increasing. Once an alias is leaked, it cannot reach the unleaked state again. On the other hand, the trust level for a sender is reversible.

One possible method to automate the process of restricting leaked aliases is to leverage existing spam technologies. For example, when an incoming mail to a particular alias is flagged by a spam filter, we automatically restrict the alias. However, given the condition of current-state-of-the-art anti-spam technologies, false positives are still possible. Thus, to avoid erroneously marking an alias as leaked, we let the user perform the marking.

5. EVALUATION

We implemented a proof-of-concept system using Postfix as the mail transfer agent and Dovecot Simple Authentication and Security Layer (SASL) for user authentication [26, 27]. We implemented the system core as Postfix advanced content filter using Python scripts. This allows us to examine and modify email headers. Frontend web scripts provide account management functions for users. We also implemented a browser extension for Firefox by modifying PwdHash [25] so that the user can request reproducible email IDs for filling out web forms.

There are four main parts to our experiments. In the first part, we present a simple case study demonstrating how a semi-private alias would be used to prevent unwanted emails from an example advertising website. In the second part, we offered the system as an option to a class that was asked to sign up with a discussion forum that requires their affiliation with the university to be validated using email addresses. Thirdly, we registered with several websites and studied potential address leakages. And lastly, we provide an analysis of the processing overhead incurred by SEAL as it forwards email messages.

5.1 Case Study

As a case study for our system for semi-private aliases, we created a new alias and registered it with the online travel website `tripadvisor.com`. This particular website was chosen for the case study because we had previously observed, while testing SEAL, that they used multiple affiliated, but unique, domains for advertising different types of offers to registered users. Hence, the pattern of emails we would anticipate receiving on an alias registered with the site intuitively serves as a good simulation of address leakage, providing a simple test of SEAL's effectiveness at providing user control. Although we acknowledge that the choice of a single website does not count as rigorous testing, we note that this case study only attempts to demonstrate SEAL's potential effectiveness at reacting to an address leak. To provide a better demonstration of SEAL's effectiveness, attempts were made to find websites or newsletters that actually leaked addresses (See Section 5.3.1). But the authors came to realize that most spam sites appear to be either quickly taken down or not reliable enough for performing experiments. Furthermore, it is unrealistic that people would actually go out of their way to register with spam sites in reality, in turn making such an experiment also somewhat unrealistic.

After registering our newly created alias with the travel website, the emails received at the alias could be classified into two broad types: those addressed from the `tripadvisor.com` domain and those addressed from some affiliate of `tripadvisor` (e.g. `cruisecritic.com`). Otherwise, no serious address leakage was observed to have occurred in the case study. Treating this as a simulated leakage — to the non-`tripadvisor` domains — intuitively, SEAL's ideal operation would be to allow the user to block all of the messages received from the affiliate domains while preserving the receipt of emails directly from `tripadvisor.com`.

All of the email received on our alias during the 40-day case study from the `tripadvisor.com` domain was addressed from either of two different email addresses, while the affiliate emails were addressed from a variety of sources, none of which featured a `tripadvisor` domain. Because of this, at any point after at least one email from each of the `tripadvisor` sources has been received by the alias, partly restricting the alias and marking any untrusted senders (those senders from the unaffiliated domains) will successfully restrict the email received to only those messages sent from `tripadvisor`. For the case study, this point occurred within six days from the time the alias was registered. Since we anticipate that address leaks to spammers would typically occur much later than this in real-world practice, the need for the user to not receive spam within the first six days of using their alias for correct functionality appears reasonable.

5.2 Affiliation Validation

To study the system being used in a real world scenario where a web service requires an official email address for validating the user's affiliation, for one semester, we provided the students of a class an option to use the system for receiving updates from a discussion forum that accepts only email addresses having university domains. Including three instructors, there were 68 potential participants. The students were neither incentivized nor disincentivized to use SEAL. They would have been able to register with the discussion forum using their academic email addresses. 55 (80.9%) people proceeded to create aliases and used SEAL actively for the whole semester. Five of the users created two aliases, one created three aliases, and another created five aliases. The others created one alias. Figure 5(a) shows the number of emails processed by

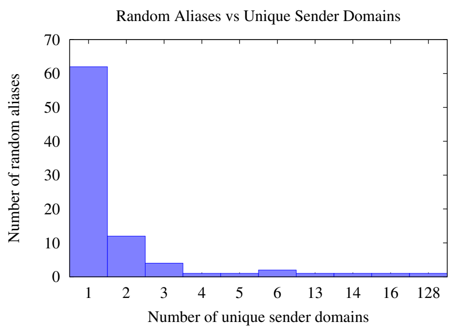


Figure 6: Histogram of the number of aliases for different number of unique sender domains.

SEAL per day while Figure 5(b) shows the daily number of aliases that were active. The days with low email transactions coincide with weekends, school break, and public holidays. While not all aliases may be active daily, the figures show that the number of users using the system remains relatively constant throughout the semester. Even though the system is only a prototype, there were no participants who stopped using the system prematurely, demonstrating the practicality of the system prototype.

5.3 Leakages

In this part of the experiments, we examine potential leakages of email aliases.

5.3.1 Leakage by Websites

To examine whether websites and mailing lists are sources of email address leakages, we created aliases and used them to register with 56 websites. In an attempt to diversify the websites instead of choosing only those ranked as highly popular by survey companies such as Alexa [28], the websites were chosen arbitrarily by searching for keywords including “shopping”, “fast cash”, “movies”, “music”, “cheap flights”, and “education”. We attempted to register with 70 web sites, succeeding in registering on 56 of them. Two of the websites initially rejected the registrations as they did not accept email ID lengths exceeding 30 characters. However, we were able to register successfully after using a shorter alias name to satisfy that requirement. Three websites disallowed the period character in email ids. We do not view this as a limitation of our system since the RFC clearly states that the email ID may be up to 64 characters long with the period character allowed [29], and also since periods are supported by prominent email services like Gmail. The remaining 10 failures were due to requiring credit card information and real cellphone numbers.

Using aliases, we also registered with another 101 websites from 15 categories listed by Alexa as the most popular sites using unique aliases [28]. The 15 categories are arts, business, computers, games, health, home, kids and teens, news, recreation, reference, regional, science, shopping, society, and sports. In addition, we subscribed to 15 mailing lists. The mailing lists were ranked as among those having the most subscribers by L-Soft, the company that invented electronic mailing lists [30].

After fifteen days, we collected and analyzed the domains from which the senders were emailing each of the aliases. We used the domains contained in the email envelopes instead of the email headers, although it is easy for an adversary to spoof either the “From” header or envelope source address. Figure 6 shows the distribution for the different numbers of aliases for varying numbers of unique sender domains. Table 3 lists examples of the sender domains. In the interest of space, only interesting cases are shown.

Table 4: Table of aliases used for classified advertising and forum postings, sorted in increasing number of unique sender domains. The layout is the same as in Table 3. Sender domains in **bold** are identified instances of leakages.

Case ID	Aliases	#	Sender Domains
B1	m4kkxa4d	1	**my.ohecampus.com**
B2	dpuxqbxg	1	**gmail.com**
B3	bob.m4kkxa4d	2	health.webmd.com, webmdmessage.com
B4	bob.n13va5ck	2	**adobe.com**, macromedia.com
B5	bob.qf11md51	2	alibaba.com, **hotmail.com**
B6	bob.wa12tfcm	8	maestro.independenttraveler.com, cruisecritic.com, tripadvisor.com, **gmail.com**, lists.sniqueaway.com, lists.airfarewatchdog.com, etc.
B7	bob.dpuxqbxg	14	**scmc050.net**, **ns2014560.ovh.net**, **kataros.com**, **fbi.gov**, **gmx.us**, **s15355439.onlinehome-server.info**, **muhleheidemusikanten.nl**, etc.

There were mails sent to 88 of the aliases. 62 (70.45%) of the aliases had senders from only one domain. Having senders from multiple domains does not necessarily constitute a leak as some domains are *affiliated*. We define two domains to be affiliated if the registrants for the domains are the same or they are declared to be affiliated in the privacy policies or terms of service. Referring to Table 3, we examined the domain affiliations for Cases A1 through A20 and identified sender domains affiliated for each alias.

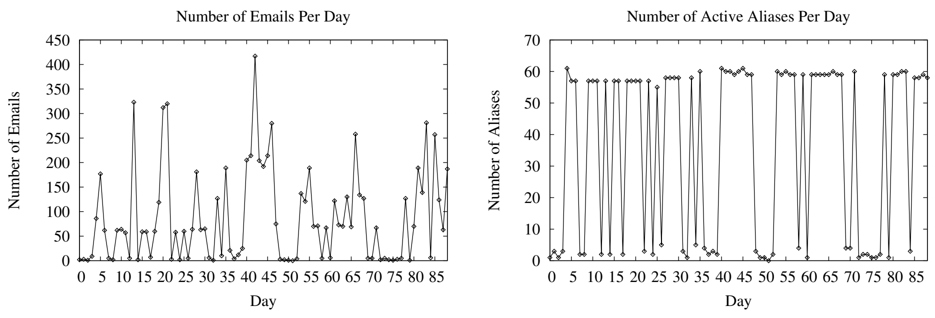
We noticed that for Cases A21 through A24, the number of unique sender domains ranged from 13 to 128. We returned to examine the private policies for these websites. All four policies stated that email addresses will be shared with other sites. An example of such a clause is, “We may share User information with third parties as reasonably necessary for us to operate this website and to provide offers and services to Users". It is easy for users to miss such clauses as they are usually obscured within lengthy privacy policies.

In practice, while investigating the sender domains, the ease with which we were able to find all emails sent to a particular alias was very encouraging. This demonstrated the advantage of using aliases for investigating potential leakages. Without using aliases, it might have been an extremely challenging task for the user to distill out bad websites such as for Cases A21 through A24. The user can then surgically mark the aliases for these cases as leaked without affecting the registrations for the good websites.

5.3.2 Leakages by Online Posts

To study the email address leakages through online message postings, we posted messages on seven forums and one popular classified advertising site. These were found from among the 45 sites we used in Section 5.3.1. For each posting, we generated a new alias and displayed it in the clear in the message body. After 15 days, we examined the mail sent to these aliases. Table 4 shows the sender domains for aliases that had emails sent to them. We observed two leakages on two forums hosted by `tripadvisor.com` and `webmd.com`. These are Cases B1, B3, and B6. Cases B1 and B3 are related. Clearly, the sender in Case B1 was intending to send to `bob.m4kkxa4d`. However, perhaps due to parsing error, the mail was sent to `m4kkxa4d` instead. We were able to observe this abnormality because we intercepted all emails sent to our server. The email in Case B1 was not forwarded to the user’s email provider. The emails sent in Cases B4 and B5 were legitimate responses to our forum postings.

In addition, there were 24 emails from various senders sent to the alias that we used for posting an advertisement on a classified advertising site. These are Cases B1 and B6 in Table 4. Based on



(a) Number of emails processed daily. (b) Number of active aliases per day.

Figure 5: SEAL usage.

Table 3: Table of aliases used for website, mailing list and newsletter registrations, sorted in increasing number of unique sender domains. The first column lists the Case IDs while the second column shows the alias. The third column indicates the number of unique domains of the senders and the last column lists some of the domains. In the interest of space, entries with senders from more than six unique domains are truncated.

Case ID	#	Sender Domains
A1	2	website.mlb.com, bounce.ed10.net
A2	2	emailconfirm.com.com, noreply.gamespot.com
A3	2	australia.care2.com, bounce.bluestatedigital.com
A4	2	connect.match.com, returnpath.bluehornet.com
A5	2	bounce.em.ign.com, bounce.mkt1839.com
A6	2	paypal.com, bounce.ed10.net
A7	2	signaturesurveys.com, server220.go-mama-hosting.com
A8	2	b.mypoints.com, mail.hpshopping.com
A9	2	envfrm.rsys5.com, animoto.com
A10	2	crosswalkmail.com, salememail.net
A11	2	email.decrease4u.net, QuickenLoans.com
A12	2	ebay.com, us.emarsys.net
A13	3	email-bounces.amazonses.com, facebookmail.com, bounce.game.e.playdom.com
A14	3	echineselearning.com, in.constantcontact.com, bmsend.com
A15	3	mail.christianmingle.com, ChristianMingle.com, believe.com
A16	3	pandaresearch.com, paidsurveysforyou.com, arcamax.com
A17	4	yourfreesurveys.com, surveyhelpcenter.com, myview.com, bounce.exacttarget.com
A18	5	rootsweb.com, email.ancestry.ca, email.ancestry.com.au, email.ancestry.com, email.ancestry.co.uk
A19	6	ssprd9.net, 5in5now.com, clearvoicesurveysmail.com, mailboto24.com, bounce.npdmr.com, mailboto21.com
A20	6	jangomail.com, ownattention.com, freebieape.info, royalofficials.com, weekenddefeat.com, galabenefits.com
A21	13	litmus.modulelaunches.net, squaresz.com, nast.zoncatalor.com, tourer.fillsavings.com, ecipwriver.com, tingly.muterdepordet.com, etc.
A22	14	downpours.net, berks.philosophersr.com, unff.neswooleston.com, brazil.lxxia.com, pinch.istrowesturase.com, paolo.flatstudio.net, etc.
A23	16	mydailymoment.com, list.cheapflights.com, inboxpays.com, bounces.lifescript.com, inboxdollars.com, mailboto21.com, etc.
A24	128	sellingprocess.info, theconfident.info, yourcouponworld.info, mycontentsite.info, mycrowdsourcecentral.info, emilestone.info, etc.

Table 5: Percentages for five groups of shortest delays.

Delay (secs)	Percentage	Number
0 to 1	81.383	103,189
1 to 2	9.465	12,001
2 to 3	1.779	2,255
3 to 4	0.900	1,141
4 to 5	0.605	767

the email contents, seven of these appear to be legitimate queries, while the remaining 17 emails contained messages that were irrelevant to the original context. Six emails claimed to be from the administrator of the site, one from FBI and one from a reputable bank. Eight of them contained links to external suspicious sites.

5.4 Timing Performance

Email systems use a store-and-forward model. Numerous factors contribute towards the time taken for an email to be sent to its recipient, including network latency, spam or virus detection filtering, and overloaded relay servers. While delays are generally tolerable, any additional processing on the emails by servers such as SEAL should be reasonable. Towards understanding the timing overheads incurred by SEAL, we measured the arrival times of emails at the email relay servers as indicated by the time stated in the `Received` header field. While this is not ideal for several reasons including clock skew between different servers, incorrect date and time on some servers, and timing information having only a granularity of seconds, it allows us to approximate the overhead incurred by SEAL. Moreover, the lack of access to other servers does not allow us a detailed comparison of performance data.

We synchronized SEAL's clock with an NTP server and assume that other servers did the same. The `Received` header fields are added by each SMTP server as the email is accepted. Figure 7 shows an example of these fields for an email. We compute the differences between the timestamps of two consecutive entries and refer to them as *delays*. In the example, we use the delay between entries 2 and 3 as an indication of the processing time required by SEAL to analyze and forward the email. At entry 2, the email is marked as received by SEAL, after which it is processed. It is then sent to the outgoing queue with entry 3 added. While we could have timed the scripts, the lack of data from other servers would render any comparison meaningless.

Table 5 shows the percentages for the five groups of the shortest delays. 126,794 delays for emails were collected from one of the author's email accounts and a SEAL account. The mean and standard deviation for these delays is 105.116 seconds and 21,232.627 seconds respectively. 3,706 delays were incurred by SEAL, with the maximum and minimum delays being five and one seconds respectively. The average delay contributed by SEAL is 0.274 seconds. This is at 0.00494 standard deviation away from the mean. This can also be inferred from Table 5 that shows the percentages for the five groups of shortest delays. 81.383% of the delays are from zero seconds to one second, in which SEAL's average delay lies. The delay incurred by SEAL is thus insignificant in comparison to other delays.

6. DISCUSSION

6.1 Security

Although our construction of semi-private aliases seeks to minimize inconvenience to legitimate senders, there are remaining issues, some of which also apply to existing DEA systems. During the transition of an alias to the restricted state, there are some cases

```
(6) by 10.231.190.83
    with SMTP id dh19csp37616ibb;
    Sat, 25 Feb 2012 07:02:20 -0800 (PST)
(5) by 10.50.178.73
    with SMTP id cw9mr7274127igc.23.1330182140761;
    Sat, 25 Feb 2012 07:02:20 -0800 (PST)
(4) from seal.eecs.umich.edu
    (d-110-235.eecs.umich.edu. [141.212.110.235])
    by mx.google.com
    with ESMTP id no10si2673927igc.10.2012.02.25.07.02.20;
    Sat, 25 Feb 2012 07:02:20 -0800 (PST)
(3) from seal.eecs.umich.edu (localhost [127.0.0.1])
    by seal.eecs.umich.edu (Postfix)
    with ESMTP id EE11954C72F
    for <johnsmith@gmail.com>;
    Sat, 25 Feb 2012 10:05:12 -0500 (EST)
(2) from backend.www.inm.smartertravel.net
    (backend.www.inm.smartertravel.net [75.98.73.172])
    by seal.eecs.umich.edu (Postfix)
    with ESMTPS id CA35354C722
    for <ads.j1pdkqa5@seal.eecs.umich.edu>;
    Sat, 25 Feb 2012 10:05:12 -0500 (EST)
(1) from smarter (helo=localhost)
    by backend.www.inm.smartertravel.net
    with local-bsmtp (Exim 4.76)
    (envelope-from
    <b-KEEXNPCTCQ-38936-2893808-AWDSubscriptionUtils
    @lists.airfarewatchdog.com>)
    id 1S1J8d-0007SB-QG
    for ads.j1pdkqa5@seal.eecs.umich.edu;
    Sat, 25 Feb 2012 10:02:19 -0500
```

Figure 7: Values of the `Received` header fields for an email, annotated with the order in which they were pushed onto the mail header. The receipt timestamps are highlighted in gray.

in which known legitimate senders may be treated as untrusted. For instance, in a more severe case, if a user is subscribed to a mailing list under a semi-private alias that the user later marks as restricted, and then the domain name of the mailing list's server is changed, the mailing list would then be treated as untrusted and would likely ignore our service's prompts to solve a CAPTCHA challenge, resulting in that newsletter being silently blocked. One simple mitigation would be to deliver these messages to the user's spam folder, instead of completely blocking them (this requires cooperation between SEAL and the mail provider). The user can then mark the senders that are incorrectly delivered as trusted.

Another concern is the potential misuse of SEAL by spammers. For example, they could create aliases to be used in the From field of spam messages, providing a channel for the recipients of spam to respond (e.g. to spam advertisements). But it is unclear if this offers significant advantages to spammers since it is trivial for them to create multiple email addresses using mail servers they control and, as far as we are aware, this does not help them bypass existing spam defense mechanisms. Note that a spammer would still have to create an account with an email provider that is coupled with their SEAL account. Legitimate SEAL servers could be configured to permit only coupling with email providers that have checks against spammer registration or receipt of large amounts of bulk mail in short intervals (e.g., Gmail appears to have such controls). Illegitimate SEAL servers that are primarily designed to protect spammers would probably get blacklisted over time, just as mail servers do.

Spammers could also attempt to attack SEAL protocols directly. For example, a spammer could attempt to spoof a legitimate user and send commands to add themselves to the trusted set. But, to do that, the spammer would need several pieces of information that are not easy to get: email ID with the mail provider and account userid/password on SEAL. Email sent to command addresses, such as getalias@sealserver, is rejected unless it arrives over an authenticated SMTP session and the commands are executed under the identity of the user that authenticated, rather than the content of the

"From" field in the message headers.

Spammers could attempt to compromise SEAL infrastructure as well. While SEAL servers should be secured using best practices, one should minimize the damage that results in case the server is compromised. We consider two forms of attacks: (1) a one-time intrusion that simply steals all the data within the databases and (2) an active attack where the attacker compromises the code within the server. In the first case, the only email IDs that the attacker gets hold of are the user's email ID at the mail provider (Gmail ID in Figure 3. All other email IDs are stored as salted hashes, which should be difficult to reverse[1]. Our implementation recommends that the user create a fresh, private account on the Gmail provider. That email address should not be publicly used – all email from it is routed via the SEAL server by configuration of SMTP settings within the mail provider. Recipients only see semi-private aliases. If the email ID at the mail provider is ever leaked, it is easy to change, since it is only relevant to the owner and not shared.

In the second case, if a spammer compromises the SEAL servers, they can monitor emails flowing through the system and collect addresses. While this is serious, the addresses collected are limited to the time that the attack goes unnoticed. It is certainly less serious than the compromise of an email provider, where both older messages and future email are potentially accessible.

SEAL is not designed to provide anonymity against local network snooping. A government, for example, could monitor the network channels to a SEAL server and collect emails, since they could go over unauthenticated and unencrypted SMTP from arbitrary senders. As far as we are aware, this is not a typical attack used by spammers.

6.2 Usability

Despite our efforts to make SEAL easy to use and minimize impact on non-spam senders, we acknowledge that some users will still prefer permanent addresses to semi-private aliases. Permanent addresses have the advantage that they can be printed on business cards, are easy to remember, and thus hand out. With SEAL, a user could generate an alias on a mobile device and then write it by hand on a form or business card (which may not be too bad for one-on-one situations). For better scalability when the user is handing out the cards to a large number of users, a possible solution would be to publish a means for a requester to send a text message and receive the alias as a response. This ties the requester's cell phone number to the alias. Cell phones are sufficiently common now among email users that we don't see this as a significant usage barrier.

For publishing email IDs on web pages, we are currently experimenting with a mechanism that generates a semi-private alias on the web page based on the IP address from which the HTTP request was received. The reason for looking into this is to investigate if this provides additional means to identify servers that are used to harvest email IDs from web sites. We are still in the process of collecting data from this mechanism.

One significant usability concern with SEAL is that, over time, one person could appear multiple times in an address book. This would occur when email containing aliases in the From or To fields is sent to a group. When those aliases are added to an address book, one person may end up with multiple aliases in an address book. This occurs today also to some extent as people both have work and personal email accounts. As a result, many address books permit multiple email IDs to be associated with one person. With SEAL, being able to mark an email ID as the preferred or primary email ID will be useful. In our design, we require the alias name of an alias to be associated with a single account. As a result, a SEAL-compatible address book could automatically associate all email IDs that have the same alias name (e.g., aliasname.*@sealserver) with the same person.

As mentioned before, the browser extension for using aliases as web usernames was adapted from PwdHash [25], and so it comes with some of the same limitations as their work, including lack of portability to all applications that render HTML, vulnerability to spyware coexisting on the same computer, and susceptibility to attacks on DNS to confuse the resolution of domain names. One potential improvement in usability over PwdHash comes from the convenient fact that the username field is not normally scrambled on login web forms, so that the user can more easily see the fetching and replacement of their username and know that it was successful. It is also notable that while the user must input a sensitive password when using PwdHash, the information being input for our extension is not nearly as sensitive, and so attacks such as focus stealing are not likely to pose as substantial a threat to web account security.

[1]Besides, if the spammer had a dictionary of email IDs, there are cheaper means of verifying them than trying to do a dictionary attack on the salted hashes.

7. CONCLUSION

The current paradigm does not provide email address owners sufficient control of their addresses, leading to email address leakages. In addition to traditional risks posed by underground crackers, some services require the users' official addresses to validate their affiliations with certain organizations. Current technologies do not allow users to provide alternative addresses that do not over-disclose user information to these services.

We propose the concept of semi-private email aliases and its embodiment, SEAL, a system that provides users more control over their email aliases and allows web services to validate the user's affiliation with an organization without having access to the user's private information. Semi-private aliases are randomized email addresses that can be restricted progressively when the user detects that they have leaked. This is related to the common disposal feature of DEA systems. However, what distinguishes SEAL from other DEA systems is that it both includes a more advanced mechanism for managing finer-grained alias lifecycles, allowing for a more flexible approach to retiring compromised email addresses, and also that it integrates fully with current email systems while at the same time not being overly restrictive. Experimental results indicate that SEAL can be useful in controlling unsolicited email, while being compatible with existing email systems.

In corporate settings, SEAL also permits use of aliases to validate a user's affiliation, while preventing disclosure of the work-related email ID or associated information. This proved useful in a test deployment where an instructor of a freshmen course at our institution required students to use an online forum provided by Piazza.com but did not wish to require the students to disclose their university email ID to the service because of concerns about student privacy. Piazza.com's default sign-up mechanism uses student's email IDs to validate their university affiliation. Over 80% of students chose to use email aliases issued by SEAL rather than their university email ID, to sign up at Piazza.com.

8. ACKNOWLEDGMENTS

This paper is based upon work supported by the National Science Foundation Grant No. CNS 0916126. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

9. REFERENCES

[1] David Mazières and M. Frans Kaashoek. The design, implementation and operation of an email pseudonym server. In *Proceedings of the 5th ACM conference on Computer and communications security*, CCS '98, pages 27–36, New York, NY, USA, 1998. ACM.

[2] Jean-Marc Seigneur and Christian Damsgaard Jensen. Privacy recovery with disposable email addresses. *IEEE Security and Privacy*, 1:35–39, November 2003.

[3] Anonymous Email: Free disposable email service for receiving emails anonymously. Online, 2011.

[4] Your Own Protection Mail. Online. `http://www.yopmail.com/en/`, 2011.

[5] Mailinator. Online, 2011.

[6] Aram Yegenian and Tassos Dimitriou. Inexpensive Email Addresses: An Email Spam-Combating System. In Sushil Jajodia, Jianying Zhou, Ozgur Akan, Paolo Bellavista, Jiannong Cao, Falko Dressler, Domenico Ferrari, Mario Gerla, Hisashi Kobayashi, Sergio Palazzo, Sartaj Sahni, Xuemin (Sherman) Shen, Mircea Stan, Jia Xiaohua, Albert Zomaya, and Geoffrey Coulson, editors, *Security and Privacy in Communication Networks*, volume 50 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 35–52. Springer Berlin Heidelberg, 2010.

[7] John Ioannidis. Fighting spam by encapsulating policy in email addresses.

[8] TMDA. Tagged Message Delivery Agent (TMDA). Online.

[9] David Recordon and Drummond Reed. Openid 2.0: a platform for user-centric identity management. In *Proceedings of the second ACM workshop on Digital identity management*, DIM '06, pages 11–16, New York, NY, USA, 2006. ACM.

[10] Gary Robinson. A statistical approach to the spam problem. *Linux J.*, 2003:3–, March 2003.

[11] Le Zhang, Jingbo Zhu, and Tianshun Yao. An evaluation of statistical spam filtering techniques. *ACM Trans. Asian Lang. Inf. Process.*, 3(4):243–269, December 2004.

[12] Ion Androutsopoulos, John Koutsias, Konstantinos Chandrinos, Georgios Paliouras, and Constantine D. Spyropoulos. An evaluation of Naive Bayesian anti-spam filtering. *CoRR*, cs.CL/0006013, 2000.

[13] Ion Androutsopoulos, John Koutsias, Konstantinos V. Chandrinos, and Constantine D. Spyropoulos. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, pages 160–167, New York, NY, USA, 2000. ACM.

[14] Karl-Michael Schneider. A comparison of event models for Naive Bayes anti-spam e-mail filtering. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, pages 307–314, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[15] Tony A. Meyer and Brendon Whateley. Spambayes: Effective open-source, bayesian based, email classification system. In *CEAS*, 2004.

[16] Cormac O'Brien and Carl Vogel. Spam filters: bayes vs. chi-squared; letters vs. words. In *Proceedings of the 1st international symposium on Information and communication technologies*, ISICT '03, pages 291–296. Trinity College Dublin, 2003.

[17] Jonathan A. Zdziarski. *Ending Spam: Bayesian Content Filtering and the Art of Statistical Language Classification*. No Starch Press, San Francisco, CA, USA, 2005.

[18] Vipul Ved Prakash and Adam O'Donnell. Fighting Spam with Reputation Systems. *Queue*, 3:36–41, November 2005.

[19] Jennifer Golbeck and James Hendler. Reputation Network Analysis for Email Filtering. In *In Proc. of the Conference on Email and Anti-Spam (CEAS), Mountain View*, 2004.

[20] P. Oscar Boykin and Vwani Roychowdhury. Personal Email Networks: An Effective Anti-Spam Tool. *IEEE COMPUTER*, 38:61, 2004.

[21] Sushant Sinha, Michael Bailey, and Farnam Jahanian. Shades of Grey: On the Effectiveness of Reputation-based "blacklists". In *Proceedings of the 3rd International Conference on Malicious and Unwanted Software (MALWARE '08)*, pages 57–64, Fairfax, Virginia, USA, October 2008.

[22] Joshua Goodman and Robert Rounthwaite. Stopping Outgoing Spam, 2004.

[23] Chris Kanich, Christian Kreibich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, Vern Paxson, and Stefan Savage. Spamalytics: an empirical analysis of spam marketing conversion. In *Proceedings of the 15th ACM conference on Computer and communications security*, CCS '08, pages 3–14, New York, NY, USA, 2008. ACM.

[24] Martín Abadi, Andrew Birrell, Mike Burrows, Frank Dabek, and Ted Wobber. Bankable Postage for Network Services. In *In Proc. Asian Computing Science Conference*, pages 72–90, 2003.

[25] Blake Ross, Collin Jackson, Nick Miyake, Dan Boneh, and John C. Mitchell. Stronger password authentication using browser extensions. In *Proceedings of the 14th conference on USENIX Security Symposium - Volume 14*, pages 2–2, Berkeley, CA, USA, 2005. USENIX Association.

[26] Postfix. Postfix. `http://www.postfix.org/`, 2011.

[27] Dovecot. Dovecot. `http://www.dovecot.org/`, 2011.

[28] Alexa. Alexa The Web Information Company. Online. `http://www.alexa.com/`, 2011.

[29] IETF. RFC 5322: Internet Message Format. Online, October 2008.

[30] Listserv. Lists with 10,000 subscribers or more. Online, July 2011.

APPENDIX

A. APPENDIX

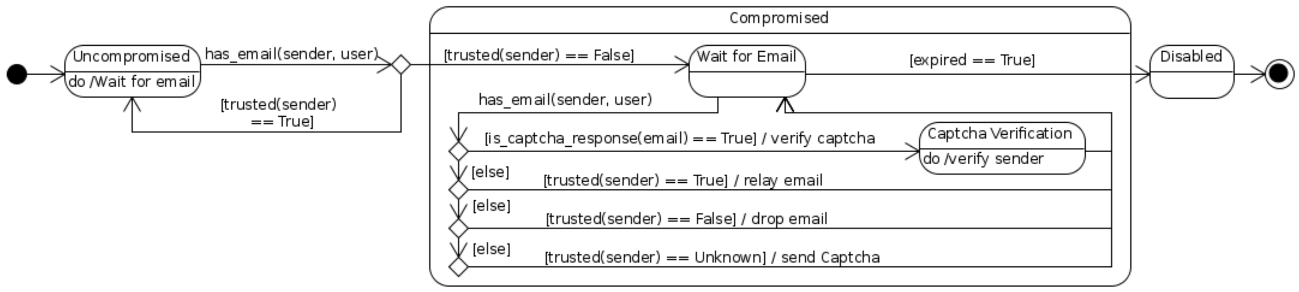

Compromised
Uncompromised
do /Wait for email
has_email(sender, user)
[trusted(sender) == True]
[trusted(sender) == False]
Wait for Email
[expired == True]
Disabled
has_email(sender, user)
[is_captcha_response(email) == True] / verify captcha
Captcha Verification
do /verify sender
[else]
[trusted(sender) == True] / relay email
[else]
[trusted(sender) == False] / drop email
[else]
[trusted(sender) == Unknown] / send Captcha

Figure 8: State diagram for alias.

```
From: bob@sealserver
To: getalias@sealserver
Subject: bobhome
```

```
From: getalias@sealserver
To: bob@sealserver
Reply-To: bobhome.b3f9cehd@sealserver
Subject: bobhome.b3f9cehd@sealserver
Body:
Your new randomized email is: "bobhome.b3f9cehd@sealserver"
Append this to your recipient list. We do not recommend
using this address for multiple recipients.
```

Figure 9: (L) Example email sent by Bob to request an alias. (R) Example response to Bob's alias request.

```
From: bob@sealserver
To: "work" alice@gmail.com
Reply-To: bob@sealserver
Subject: Business Proposal
Body:
Dear Alice, ...
```

Figure 10: Example of using *hint*.