Detecting Credential Spearphishing Attacks in Enterprise Settings

Grant Ho[†] Aashish Sharma[◇] Mobin Javed[†] Vern Paxson[†*] David Wagner[†]

[†]UC Berkeley [◇]Lawrence Berkeley National Laboratory [*]International Computer Science Institute

Abstract

We present a new approach for detecting credential spearphishing attacks in enterprise settings. Our method uses features derived from an analysis of fundamental characteristics of spearphishing attacks, combined with a new non-parametric anomaly scoring technique for ranking alerts. We evaluate our technique on a multi-year dataset of over 370 million emails from a large enterprise with thousands of employees. Our system successfully detects 6 known spearphishing campaigns that succeeded (missing one instance); an additional 9 that failed; plus 2 successful spearphishing attacks that were previously unknown, thus demonstrating the value of our approach. We also establish that our detector's false positive rate is low enough to be practical: on average, a single analyst can investigate an entire month's worth of alerts in under 15 minutes. Comparing our anomaly scoring method against standard anomaly detection techniques, we find that standard techniques using the same features would need to generate at least 9 times as many alerts as our method to detect the same number of attacks.

1 Introduction

Over the past several years, a litany of high-profile breaches has highlighted the growing prevalence and potency of spearphishing attacks. Leveraging these attacks, adversaries have successfully compromised a wide range of government systems (e.g., the US State Department and the White House [1]), prominent companies (e.g., Google and RSA [3]), and recently, political figures and organizations (e.g., John Podesta and the DNC [21]).

Unlike exploits that target technical vulnerabilities in software and protocols, spearphishing is a type of social engineering attack where the attacker sends a targeted, deceptive email that tricks the recipient into performing some kind of dangerous action for the adversary. From an attacker's perspective, spearphishing requires little technical sophistication, does not rely upon any specific vulnerability, eludes technical defenses, and often succeeds. From a defender's perspective, spearphishing is difficult to counter due to email's susceptibility to spoofing and because attackers thoughtfully handcraft their attack emails to appear legitimate. For these reasons, there are currently no generally effective tools for detecting or preventing spearphishing, making it the predominant attack for breaching valuable targets [17].

Spearphishing attacks take several forms. One of the most well-known involves an email that tries to fool the recipient into opening a malicious attachment. However, in our work, which draws upon several years worth of data from the Lawrence Berkeley National Lab (LBNL), a large national lab supported by the US Department of Energy, none of the successful spearphishing attacks involved a malicious attachment. Instead, the predominant form of spearphishing that LBNL encounters is *credential spearphishing*, where a malicious email convinces the recipient to click on a link and then enter their credentials on the resulting webpage. For an attachment-driven spearphish to succeed against a site like LBNL, which aggressively scans emails for malware, maintains frequently updated machines, and has a team of several full-time security staff members, an attacker will often need to resort to an expensive zero-day exploit. In contrast, credential spearphishing has an incredibly low barrier to entry: an attacker only needs to host a website and craft a deceptive email for the attack to succeed. Moreover, with widespread usage of remote desktops, VPN applications, and cloud-based email providers, stolen credentials often provide attackers with rich information and capabilities. Thus, although other forms of spearphishing constitute an important threat, credential spearphishing poses a major and unsolved threat in-and-of itself.

Our work presents a new approach for detecting credential spearphishing attacks in enterprise settings. This domain proves highly challenging due to base-rate issues. For example, our enterprise dataset contains 370 million emails, but fewer than 10 known instances of spearphishing. Consequently, many natural methods fail, because their false positive rates are too high: even a false positive rate as low as 0.1% would lead to 370,000 false alarms. Additionally, with such a small number of known spearphishing instances, standard machine learning approaches seem unlikely to succeed: the training set is too small and the class imbalance too extreme.

To overcome these challenges, we introduce two key contributions. First, we present an analysis of characteristics that we argue are fundamental to spearphishing attacks; from this analysis, we derive a set of features that target the different stages of a successful spearphishing attack. Second, we introduce a simple, new anomaly detection technique (called *DAS*) that requires no labeled training data and operates in a non-parametric fashion. Our technique allows its user to easily incorporate domain knowledge about their problem space into the anomaly scores DAS assigns to events. As such, in our setting, DAS can achieve an order-of-magnitude better performance than standard anomaly detection techniques that use the same features. Combining these two ideas together, we present the design of a real-time detector for credential spearphishing attacks.

Working with the security team at LBNL, we evaluated our detector on nearly 4 years worth of email data (about 370 million emails), as well as associated HTTP logs. On this large-scale, real-world dataset, our detector generates an average of under 10 alerts per day; and on average, an analyst can process a month's worth of these alerts in 15 minutes. Assessing our detector's true positive accuracy, we find that it not only detects all but one spearphishing attack known to LBNL, but also uncovers 2 *previously undiscovered* spearphishing attacks. Ultimately, our detector's ability to identify both known and novel attacks, and the low volume and burden of alerts it imposes, suggests that our approach provides a practical path towards detecting credential spearphishing attacks.

2 Attack Taxonomy and Security Model

In a spearphishing attack, the adversary sends a targeted email designed to trick the recipient into performing a dangerous action. Whereas regular phishing emails primarily aim to make money by deceiving any arbitrary user [18, 22], spearphishing attacks are *specifically targeted* at users who possess some kind of *privileged access or capability* that the adversary seeks. This selective targeting and motivation delineates spearphishing (our work's focus) from regular phishing attacks.

2.1 Taxonomy for Spearphishing Attacks

Spearphishing spans a wide range of social-engineering attacks. To better understand this complex problem space, we present a taxonomy that characterizes spearphishing attacks across two dimensions. These correspond to the two key stages of a successful attack. Throughout this paper, we refer to the attacker as Mallory and the victim as Alice.

2.1.1 Lure

Spearphishing attacks require Mallory to convince Alice to perform some action described in the email. To accomplish this, Mallory needs to imbue her email with a sense of trust or authority that convinces Alice to execute the action. Attackers typically achieve this by sending the email under the identity of a trusted or authoritative entity and then including some compelling content in the email.

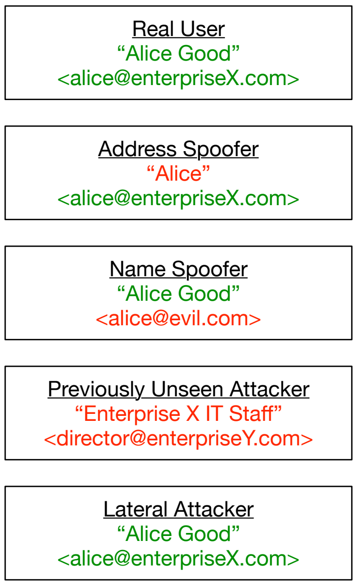


Figure 1: Examples of four different impersonation models for a real user "Alice Good". In the *address spoofer* impersonation model, an attacker might also spoof the username to exactly match the true user's (e.g., by using `Alice Good` instead of just `Alice`). Our work focuses on detecting the latter three threat models, as discussed in Section 2.2: *name spoofer*, *previously unseen attacker*, and *lateral attacker*.

Impersonation Model: Spearphishing involves impersonating the identity of someone else, both to create trust in the recipient and also to to minimize the risk of attribution and punishment. There are several types of impersonation:

1. An *address spoofer* uses the email address of a trusted individual in the `From` field of the attack email. The attacker may spoof the name in the `From` header as well, so that the attacker's `From` header exactly matches the true user's typical `From` header.
   DKIM and DMARC [2] block this impersonation model by allowing domains to sign their sent emails' headers with a cryptographic signature, which receiving servers can verify with a DNS-based verification key. In recent years, these protocols have seen increasingly widespread adoption, with many large email providers, such as Gmail, deploying them in response to the rise of phishing attacks [4].

2. A *name spoofer* spoofs the name in their email's `From` header to exactly match the name of an existing, trusted individual (e.g.,`Alice Good` in `Alice Good <alice@evil.com>`). However, in this impersonation model, the attacker does not forge the email address of their `From` header, relying instead on the recipient to only view the name of the sender, or on the recipient's mail client to show only the name of the sender. By not spoofing the `From` email address, this impersonation model circumvents DKIM/DMARC.
3. A *previously unseen attacker* selects a name and email address to put in the `From` field of the spearphishing email, where neither the name nor the email address actually match a true user's name or email address (though they might be perceived as trustworthy or similar to a real user's identity). For instance, Mallory might choose to spoof the name `LBNL IT Staff` and the email address `<helpdesk@enterpriseY.com>`.
4. A *lateral attacker* sends the spearphishing email from a compromised user's email account.

2.1.2 Exploit Payload

Once Mallory has gained Alice's trust, she then needs to exploit this trust by inducing Alice to perform some dangerous action. Three types of exploitation are commonly seen: (i) attachments or URLs that contain malware, (ii) URLs leading to websites that attempt to trick Alice into revealing her credentials, and (iii) out-of-band actions (e.g., tricking a company's CFO into wiring money to a fake, malicious "corporate partner").

2.2 Security Model

Threat Model: In this work, we specifically focus on an enterprise *credential spearphishing* threat model, where Mallory tries to fool a targeted enterprise's user (Alice) into revealing her credentials. We assume that the adversary can send arbitrary emails to the victim and can convince the recipient to click on URLs embedded in the adversary's email (leading the victim to a credential phishing website). To impersonate a trusted entity, the attacker may set any of the email header fields to arbitrary values.

In other words, we focus on attacks where Mallory's lure includes masquerading as a trusted entity, her payload is a link to a credential phishing page, and she chooses from any of the last three impersonation models. Because organizations can deploy DKIM/DMARC to mitigate address spoofing (and many large email providers have done so), we exclude address spoofing from our work.

Security Goals: First, a detector must produce an extremely low false positive burden, ideally only 10 or so false alarms per day that take at most minutes for an incident response team to process. Second, a detector must detect real spearphishing attacks (true positives). Given that current methods for detecting credential spearphishing often rely on users to report an attack, if our approach can detect even a moderate number of true positives or identify undiscovered attacks, while achieving a low false positive rate, then it already serves as a major improvement to the current state of detection and mitigation.

Data Source	Fields/Information per Entry
SMTP logs	Timestamp `From` (sender, as displayed to recipient) `RCPT TO` (all recipients; from the SMTP dialog)
NIDS logs	URL visited SMTP log id for the earliest email with this URL Earliest time this URL was visited in HTTP traffic # prior HTTP visits to this URL # prior HTTP visits to any URL with this hostname Clicked hostname (fully qualified domain of this URL) Earliest time any URL with this hostname was visited
LDAP logs	Employee's email address Time of current login Time of subsequent login, if any # total logins by this employee # employees who have logged in from current login's city # prior logins by this employee from current login's city

Table 1: Schema for each entry in our data sources. All sensitive information is anonymized before we receive the logs. The NIDS logs contain one entry for each visit to a URL seen in any email. The LDAP logs contain one entry for each login where an employee authenticated from an IP address that he/she has never used in prior (successful) logins.

3 Datasets

Our work draws on the SMTP logs, NIDS logs, and LDAP logs from LBNL; several full-time security staff members maintain these extensive, multi-year logs, as well as a well-documented incident database of successful attacks that we draw upon for our evaluation in Section 6. For privacy reasons, before giving us access to the data, staff members at LBNL anonymized all data using the procedure described in each subsection below. Additionally, our anonymized datasets do not contain the contents of email bodies or webpages. Table 1 shows the relevant information in these datasets and Table 2 summarizes the size and timeframe of our data.

3.1 SMTP Logs

The SMTP logs contain anonymized SMTP headers for all inbound and outbound emails during the Mar 1, 2013 – Jan 14, 2017 time period. These logs contain information about all emails sent to and from the organization's employees (including emails between two employees), a total of 372,530,595 emails. The second row of Table 1 shows the relevant header information we receive for each email in these logs.

The data was anonymized by applying a keyed hash to each sensitive field. Consider a header such as `Alice Good <ali@company.com>`. The 'name' of a header is the human name (`Alice Good` in our example); when no human name is present, we treat the email address as the header's 'name'. The 'address' of a header is the email address: `<ali@company.com>`. Each name and each email address is separately hashed.

3.2 NIDS Logs

LBNL has a distributed network monitor (Bro) that logs all HTTP GET and POST requests that traverse its borders. Each log entry records information about the request, including the full URL.

Additionally, the NIDS remembers all URLs seen in the bodies of inbound and outbound emails at LBNL.[1] Each time any URL embedded in an email gets visited as the destination of an HTTP request, the NIDS will record information about the request, including the URL that was visited and the entry in the SMTP logs for the email that contained the fetched URL. The NIDS remembers URLs for at least one month after an email's arrival; all HTTP visits to a URL are matched to the earliest email that contained the URL.

We received anonymized logs of all HTTP requests, with a keyed hash applied to each separate field. Also, we received anonymized logs that identify each email whose URL was clicked, and anonymized information about the email and the URL, as shown in Table 1.

3.3 LDAP Logs

LBNL uses corporate Gmail to manage its employees' emails.[2] Each time an employee successfully logs in, Gmail logs the user's corporate email address, the time when the login occurred, and the IP address from which the user authenticated. From these LDAP logs, we received anonymized information about login sessions where (1) the login IP address had never been used by the user during any previous successful login, (2) the user had more than 25 prior logins, and (3) the login IP address did not belong to LBNL's network. The last row of Table 1 shows the anonymized data in each entry of the LDAP logs.

4 Challenge: Diversity of Benign Behavior

Prior work has used machine learning to identify spearphishing attacks, based on suspicious content in email headers and bodies [8,19]. While that work detects several spearphishing attacks, their optimal false positive rates (FPR) are 1% or higher, which is far too high for our setting: a FPR of 1% would lead to 3.7 million false alarms on our dataset of nearly 370 million.

In this section, we identify several issues that make spearphishing detection a particularly difficult challenge. Specifically, when operating on a real-world volume of millions of emails per week, the diversity of benign behavior produces an untenable number of false positives for detectors that merely look for anomalous header values.

[1]Shortened URLs are expanded to their final destination URLs.

[2]Email between two employees also flows through corporate Gmail, which allows our detector to scan "internal" emails for lateral spearphishing attacks.

Time span	Mar 1, 2013– Jan 14, 2017
Total emails	372,530,595
Unique sender names (names in `From`)	3,415,471
Unique sender addresses (email addresses in `From`)	4,791,624
Emails with clicked URLs	2,032,921
Unique sender names (names in `From`)	246,505
Unique sender addresses (email addresses in `From`)	227,869
# total clicks on embedded URLs	30,011,810
Unique URLs	4,014,412
Unique hostnames	220,932
Logins from new IP address	219,027
# geolocated cities among all new IP addresses	7,937
# of emails sent during sessions where employee logged in from new IP address	2,225,050

Table 2: Summary of data in the three logs. Note that some emails contain multiple URLs, some or all of which may be visited multiple times by multiple recipients (thus, there are more clicked-URLs than emails that contain clicked-URLs).

4.1 Challenge 1: Senders with Limited Prior History

A natural detection strategy is to compare the headers of the current email under analysis against all historical email headers from the current email's purported sender. For example, consider a *name spoofer* who attempts to spearphish one of Alice's team members by sending an email with a `From` header of `Alice Good <alice@evil.com>`. An anomaly-based detector could identify this attack by comparing the email's `From` address (`<alice@evil.com>`) against all `From` addresses in prior email with a `From` name of `Alice Good`.

However, this approach will not detect a different spearphishing attack where neither the name nor the address of the `From` header have ever been seen before: `Alice <alice@evil.com>` or `HR Team <hr.enterpriseX@gmail.com>`. In this *previously unseen attacker* setting, there is no prior history to determine whether the `From` address is anomalous.

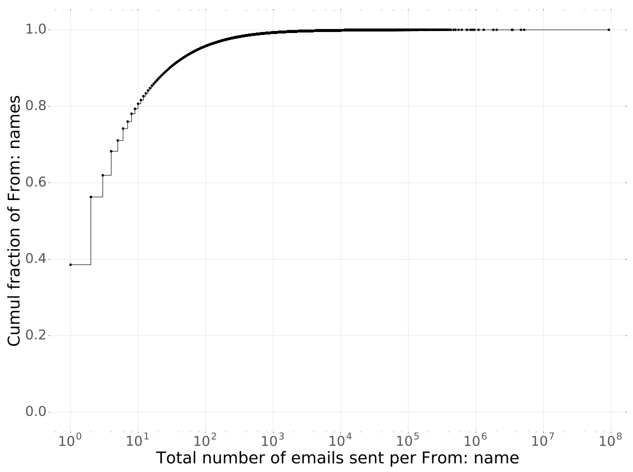


Figure 2: Distribution of the number of emails sent per `From` name. Nearly 40% of all `From` names appear in only one email and over 60% of all `From` names appear in three or fewer emails.

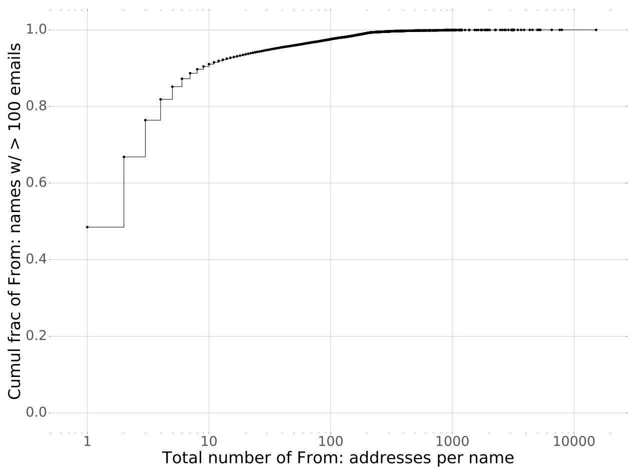


Figure 3: Distribution of the total number of `From` addresses per `From` name (who send over 100 emails) across all emails sent by the `From` name. Over half (52%) of these `From` names sent email from two or more `From` addresses (i.e., have at least one new `From` address).

To address this gap, one might flag all emails with a new or previously unknown `From` name (e.g., any email where the `From` name has been seen in two or fewer emails leads to an alert). Unfortunately, this approach generates an overwhelming number of alerts in practice because millions of `From` names are only ever seen in a few emails. Figure 2 shows the distribution of the number of emails per `From` name in our dataset. In particular, we find that over 60% of `From` names sent three or fewer emails and over 40% of `From` names sent exactly one email. Thus, even if one ran a detector retrospectively to alert on every email with a `From` name that had never been seen before and did not eventually become an active and engaged sender, it would produce over 1.1 million alerts: a false positive rate of less than 1% on our dataset of nearly 370 million emails, but still orders of magnitude more than our target. Even though spam might account for a proportion of these emails with new `From` names, LBNL's security staff investigated a random sample of these emails and found a spectrum of benign behavior: event/conference invitations, mailing list management notices, trial software advertisements, and help support emails. Thus, a detector that only leverages the traditional approach of searching for anomalies in header values faces a stifling range of anomalous but benign behavior.

4.2 Challenge 2: Churn in Header Values

Even if we were to give up on detecting attacks that come from previously unseen `From` names or addresses, a detector based on header anomalies still runs into yet another spectrum of diverse, benign behavior. Namely, header values for a sender often change for a variety of benign reasons. To illustrate this, we consider all `From` names that appear in at least 100 emails (our dataset contains 125,172 of them) and assess the frequency at which these names use a new `From` email address when sending email.

Figure 3 shows the cumulative distribution of the total number of `From` email addresses per `From` name. From this graph, we see that even among `From` names with substantial history (sent over 100 emails), there is considerable variability in header values: 52% of these `From` names send email from more than one `From` email address. We find that 1,347,744 emails contain a new `From` email address which has never been used in any of the `From` name's prior emails. Generating an alert for each of these emails would far exceed our target of 10 alerts per day.

This large number of new email addresses per `From` name stems from a variety of different sources: work vs. personal email addresses for a user, popular human names where each email address represents a different person in real life (e.g., multiple people named `John Smith`), professional society surveys, and functionality-specific email addresses (e.g. `Foo <noreply@foo.com>`, `Foo <help@foo.com>`, `Foo <donate@foo.com>`). While it might be tempting to leverage domain reputation or domain similarity between a new `From` address and the `From` name's prior addresses to filter out false positives, this fails in a number of different cases. For example, consider the case where `Alice` suddenly sends email from a new email address, whose domain is a large email hosting provider; this could either correspond to Alice sending email from her personal email account, or it might represent a *name spoofer* using a Gmail account with a spoofed `From` name.

Given the prevalence of emails with anomalous, yet benign, header values, a practical detector clearly needs to leverage additional signals beyond an email's header values. Some prior academic work has attempted to incorporate stylometry features from an email's body to identify spearphishing attacks [19]; however, as discussed earlier, these systems have false positive rates of 1% or higher, which would lead to millions of false alarms, a prohibitively high number for practical usage. In the following section, we present a novel approach that leverages a different set of signals based on the underlying nature of spearphishing attacks.

5 Detector Design

At a high level, our detector consists of three stages illustrated in Figure 4 and described below: a feature extraction stage (§ 5.1 and § 5.2), a nightly scoring stage (§ 5.4), and a real-time alert generation stage (§ 5.5). Conceptually, our work introduces two key ideas that enable our detector to detect a wide range of attacks, while achieving a practical volume of false positives that is over 200 times lower than prior work. First, our detector extracts two sets of reputation-based features that independently target the two key stages of a spearphishing attack identified in our attack taxonomy. Second, we introduce a novel, unsupervised anomaly detection technique that enables our detector to automatically rank a set of unlabeled events and select the most suspicious events for the security team to review. We first discuss each of these elements and then show how to combine them for our real-time detector.

5.1 Features per Attack Stage

Fundamentally, spearphishing attacks aim to trick their recipients into performing a dangerous action described in the email. If the attacker fails to persuade the victim into taking the action, the attack fails. For credential spearphishing, the dangerous action is clicking on a link in an email that leads the victim to a credential phishing website.[3] Thus, we analyze every email that contains a link that a user clicked on; we call this clicked link a *click-in-email* event.

As discussed in our taxonomy (§ 2.1), spearphishing attacks consist of two necessary stages: the lure stage, where the attacker persuades the victim to trust him, and the exploit stage, where the victim performs a dangerous action for the attacker. This insight leads to the first core idea in our approach: we craft two sets of features to target both of these stages of a spearphishing attack. Prior work has often used features that capture only the lure or the exploit; our insight is that we can do significantly better by using both types of features.

Accordingly, we have two classes of features: *domain reputation* features, and *sender reputation* features. In order to steal the victim's credentials, the attacker must link to a site under her control. Because spearphishing attacks are so tightly targeted, visits to this malicious website will presumably be rare among the historical network traffic from the organization's employees. Therefore, for each click-in-email event, the domain reputation features characterize the likelihood that an employee would visit that URL, based on its (fully qualified) domain. The sender reputation features characterize whether the sender of that email falls under one of the impersonation models outlined in our taxonomy. Effectively, the sender reputation features capture elements of the lure (by recognizing different types of spoofing that the attacker might use to gain the victim's trust), and the domain reputation features capture characteristics of the exploit.

Because the sender reputation features differ for each impersonation model (§ 5.2.2), our detector actually consists of three sub-detectors, one for each impersonation model. As discussed below (§ 5.5), if any of the sub-detectors flags an email as spearphishing, the detector treats it as an attack and generates an alert for the security team.

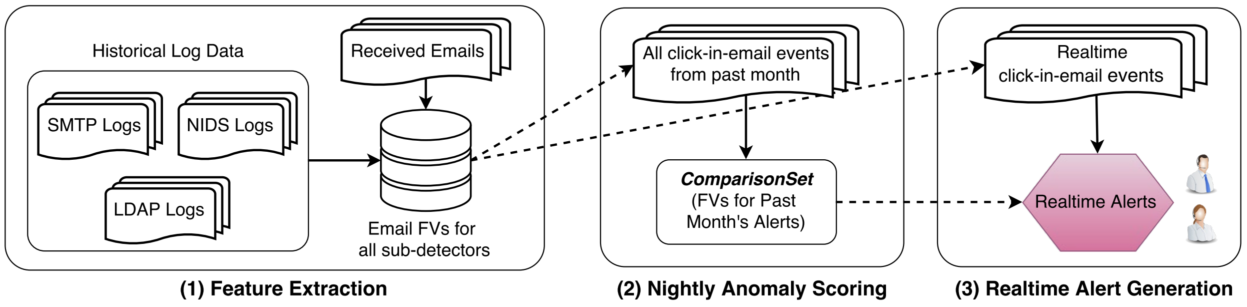
5.2 Features

Each sub-detector uses a feature vector containing four scalar values, two for domain reputation and two for sender reputation; Appendix A contains a summary table of these features, which we discuss below. As we show later (§ 6), these compact feature vectors suffice to detect a wide-range of attacks while achieving a practical volume of false positives.

5.2.1 Domain Reputation Features

All sub-detectors use the same two features to characterize the reputation of a link that the user clicked on. Intuitively, if few employees from the enterprise have visited URLs from the link's domain, then we would like to treat a visit to the email's link as suspicious. Additionally, if employees have never visited URLs from a domain until very recently, then we would also like to treat visits to the domain's URLs as risky. Based on these two ideas, the first feature counts the number of prior visits to any URL with the same fully qualified domain name (FQDN) as the clicked URL; this is a global count across all employees' visits, from the NIDS logs. The second fea-

[3] While an adversary could attempt to spearphish an employee's credentials by fooling them into including the credentials in an email response, this attack variant is likely more difficult to successfully execute given employee awareness from security training and education. Based on their multi-year incident database, LBNL has not observed such attacks succeed in practice.



Historical Log Data
SMTP Logs
NIDS Logs
LDAP Logs
Received Emails
Email FVs for all sub-detectors
All click-in-email events from past month
ComparisonSet (FVs for Past Month's Alerts)
Realtime click-in-email events
Realtime Alerts
(1) Feature Extraction
(2) Nightly Anomaly Scoring
(3) Realtime Alert Generation

Figure 4: Overview of our real-time detector, which leverages output from a nightly batch job during its real-time analysis, as described in § 5.4 and § 5.5. As emails arrive, our detector leverages historical logs to extract and save three feature vectors (one FV per impersonation model) for each URL in the email (§ 5.1). Using the network traffic logs, our detector logs all clicks on URLs embedded in emails. Each night, our detector runs our anomaly scoring algorithm on the FVs from a sliding window over the past month's clicked URLs and stores a *ComparisonSet* of the month's most suspicious FVs for each impersonation model (§ 5.4). Observing real-time network traffic, our detector sees clicked email URLs, compares the real-time click's feature vector for each impersonation model against the *ComparisonSet*, and generates an alert for the security team if needed (§ 5.5).

ture counts the number of days between the first visit by any employee to a URL on the clicked link's FQDN and the time when the clicked link's email initially arrived at LBNL.

We chose to characterize a clicked link's reputation in terms of its FQDN, rather than the full URL, because over half of the clicked URLs in our dataset had never been visited prior to a click-in-email event. Consequently, operating at the granularity of the full URL would render the URL reputation features ineffective because the majority of URLs would have the lowest possible feature values (i.e., never been visited prior to the email recipient). Additionally, using a coarser granularity such as the URL's registered domain name or its effective second-level domain could allow attackers to acquire high reputation attack URLs by hosting their phishing webpages on popular hosting sites (e.g., attacker.blogspot.com). By defining a URL's reputation in terms of its FQDN, we mitigate this risk.

5.2.2 Sender Reputation Features

Name Spoofer: As discussed earlier (§ 2.1.1), in this attacker model Mallory masquerades as a trusted entity by spoofing the name in the `From` header, but she does not spoof the name's true email address. Because the trusted user that Mallory impersonates does not send email from Mallory's spoofed address, the spearphishing email will have a `From` email address that does not match any of the historical email addresses for its `From` name. Therefore, the first sender reputation feature counts the number of previous days where we saw an email whose `From` header contains the same name and address as the email being scored.

Also, in this attacker model, the adversary spoofs the `From` name because the name corresponds to someone known and trusted. If that name did not correspond to someone trustworthy or authoritative, there would be no point in spoofing it, or it would manifest itself under our *previously unseen attacker* threat model. Thus, the second sender reputation feature for a clicked email reflects the trustworthiness of the name in its `From` header. We measure the trustworthiness of a name by counting the total number of weeks where this name sent at least one email for every weekday of the week. Intuitively, the idea is that `From` names that frequently and consistently send emails will be perceived as familiar and trustworthy.

Previously Unseen Attacker: In this threat model (§ 2.1.1), Mallory chooses a name and email address that resembles a known or authoritative entity, but where the name and email address do not exactly match any existing entity's values (e.g.,`IT Support Team <helpdesk@company.`*`net`*`>`); if the name or address did exactly match an existing entity, the attack would instead fall under the *name spoofer* or *address spoofer* threat model. Compared to name-spoofing attacks, these attacks are more difficult to detect because we have no prior history to compare against; indeed, prior work does not attempt to detect attacks from this threat model. To deal with this obstacle, we rely on an assumption that the attacker will seek to avoid detection, and thus the spoofed identity will be infrequently used; each time Mallory uses the spoofed identity, she runs the risk that the employee she's interacting with might recognize that Mallory has forged the name or email address and report it. Accordingly, we use two features: the number of prior days that the `From` name has sent email, and the number of prior days that the `From` address has sent emails.

Lateral Attacker: This sub-detector aims to catch spearphishing emails sent from a compromised user's accounts (without using any spoofing). To detect this powerful class of attackers, we leverage the LDAP logs provided by Gmail's corporate email services (§ 3). When a recipient clicks on a link in an email, if the email was sent by an employee, we check the LDAP logs to see if the email was sent during a login session where the sender-employee logged in using an IP address that the sender-employee has never used before. If so, this sub-detector computes the geolocated city of the session's IP address, say city C. It then extracts two features: the number of distinct employees that have logged in from city C, and the number of previous logins where this sender-employee logged in from an IP address that geolocated to city C.

Content-Based Features: As discussed in Section 3, for privacy reasons we do not have access to either the bodies of emails or the contents of a clicked URL's webpage. If desired, enterprises could augment our sender reputation features with additional features from the raw content in the email message or website (e.g., NLP features that characterize whether the email message relates to accounts/credentials/passwords or reflects particular sentiments such as urgency).

5.3 Limitations of Standard Detection Techniques

Once our detector has extracted features for each click-in-email event, it needs to decide which ones should trigger an alert for the security team. We first discuss three natural, but ultimately ineffective, approaches for determining which events to alert on. Then, in the following subsection, we present a new technique that our detector uses to overcome the limitations of these canonical approaches.

Manual Thresholds: The simplest approach would be to manually select a threshold for each feature, and generate an alert if all feature values are below the threshold. One might use domain knowledge of each feature to guess a threshold for each feature dimension: e.g., spearphishing attacks will use URLs whose domain has fewer than five visits or was first visited less than five days ago. Unfortunately, this approach is inherently arbitrary since we do not know the true distribution of feature values for spearphishing attacks. Thus, this ad hoc approach can easily miss attacks, and does not provide a selection criteria that generalizes across different enterprises.

Supervised Learning: A large body of literature on attack detection, from spam classification to prior spearphishing work, draws heavily on supervised machine learning algorithms. However, those methods are not suitable for our setting.

To accurately classify new events, supervised learning techniques require a labeled training dataset that reflects the range of possible malicious and benign feature values. Unfortunately, in our context, it is difficult to assemble a large enough training set. Because spearphishing attacks are extremely difficult to detect and occur at a low rate, we have few malicious samples to train on.

Additionally, our setting exhibits extreme class imbalance: because of the scarcity of data on known spearphishing attacks, the training set will have vastly more benign instances than malicious instances. Supervised techniques often need a relatively balanced dataset; classifiers trained on highly imbalanced data often learn to always predict the majority class (missing real attacks), pathologically overfit to accidental characteristics of the minority class, or generate too lax of a decision boundary and generate prohibitively high numbers of false positives [10]. While the machine learning community has explored a number of techniques for addressing imbalanced training data [6, 10], such as undersampling the over-represented class or synthetically generating samples for the under-represented class, these techniques do not scale to imbalances on the order of millions to one.

Standard Anomaly Detection: Alternatively, one might consider unsupervised or semi-supervised anomaly detection techniques. While a number of such techniques exist, including density estimation techniques such as Gaussian Mixture Models (GMMs) [5] and clustering and distance-based techniques such as k-nearest-neighbor (kNN) [13], these classical techniques suffer from three limitations.

First, in a number of security settings, scalar features often have a directionality to their values; and indeed, all of our features have this property. For example, the fewer visits a domain has, the more suspicious it is; an unusually small number of visits is grounds for suspicion, but an unusually large number is not. Standard anomaly detection techniques do not incorporate notions of asymmetry or directionality into their computations. For example, density-based anomaly detection techniques such as kernel density estimation (KDE) and GMMs fit a probability distribution to the data and alert on the lowest-probability events. Events that have statistically extreme—but benign—feature values will have a very low probability of occurring, triggering a large number of useless alerts.

Second, standard anomaly detection techniques often treat an event as anomalous even if only one or a few of the event's features are statistically anomalous. However, in our setting, we expect that attacks will be anomalous and suspicious in *all* feature dimensions. Consequently, in our setting, classical techniques will generate

Algorithm 1 Scoring and Alert Selection in DAS

```
   Score($E$, $L$):
1: for each event $X$ in $L$ do:
2:     if $E$ is more suspicious than $X$ in every dimension:
3:         Increment $E$'s score by one

   AlertGen($L$ (a list of events), $N$):
1: for each event $E$ in $L$ do:
2:     Score($E$, $L$)
3: Sort $L$ by each event's score
4: return the $N$ events from $L$ with the highest scores
```

many spurious alerts for events that are only anomalous in a few dimensions. As we show in Section 6.3, this causes classical techniques to miss the vast majority of spearphishing attacks in our dataset because they exhaust their alert budget with emails that have benign feature values in all but one dimension.

Third, classical techniques are parametric: they either assume the data comes from a particular underlying distribution, or they contain a number of parameters that must be correctly set by their deployer in order for the technique to obtain acceptable performance. GMMs assume the data comes from a mixture of Gaussian distributions, KDE has a bandwidth parameter that requires tuning by the deployer, and kNN needs the deployer to select a value of k (the number of nearest neighbors/most similar events, which the algorithm will use to compute an event's anomaly score). These requirements are problematic for spearphishing detection since we do not know the true distribution of attack and benign emails, the underlying distribution might not be Gaussian, and we do not have a sound way to select the parameters.

5.4 Directed Anomaly Scoring (DAS)

Given the limitations of traditional detection techniques, we introduce a simple and general technique for automatically selecting the most suspicious events from an unlabeled dataset. We call our technique Directed Anomaly Scoring (DAS). At a high level, DAS ranks all events by comparing how suspicious each event is relative to all other events. Once all events have been ranked, DAS simply selects the N most suspicious (highest-ranked) events, where N is the security team's alert budget.

Algorithm 1 shows the procedure for scoring and generating alerts with DAS. Concretely, DAS first assigns an anomaly score for each event, E, by computing the total number of other events where E's feature vector is at least as *suspicious* as the other event in *every* feature dimension. Thus, E's score counts how many events it is at least as suspicious as; events with higher scores are more suspicious than ones with lower scores. Figure 5 presents a few visual examples of computing DAS scores. After scoring every event, our algorithm simply sorts all events by their scores and outputs the N highest-scoring events.

Formally, we identify each event with its feature vector $E \in \mathbb{R}^d$. We consider event E to be at least as suspicious as event E', written $E \succcurlyeq E'$, if $E_i \leq E'_i$ for all $i = 1, 2, \ldots, d$. (For simplicity, we assume that smaller feature values are more suspicious, in every dimension; for dimensions where the reverse is true, we replace the comparator $\leq$ with $\geq$. Appendix A summarizes the comparators we use for each feature.) Then, the score of event E is the cardinality of the set $\{E' : E \succcurlyeq E'\}$.

DAS is well-suited for a range of security detection problems where attacks can be characterized by a combination of numerical and boolean features, such as our spearphishing use case. As we show in Section 6, DAS achieves orders-of-magnitude better results than classical anomaly detection techniques because it leverages domain knowledge about which regions of the feature space are most suspicious; in particular, it overcomes all three limitations of classical techniques discussed in Section 5.3.

5.5 Real-time Detection Architecture

We now synthesize the ideas discussed in previous subsections to provide an end-to-end overview of how we leverage DAS to generate alerts (illustrated in Figure 4). Our detector has access to the enterprise's log data, real-time network traffic (e.g., via a NIDS like Bro), and an alert budget β for each sub-detector, which specifies the daily volume of alerts that the security team deems acceptable. As each email arrives, for each URL in the email, our detector extracts the feature vector for that URL and saves it in a table indexed by the URL. Each HTTP request seen by the enterprise's NIDS is looked up in the table. Each time the detector sees a visit to a URL that was earlier seen in some email (a "click-in-email event"), it adds that feature vector to a list of events. Finally, our detector uses the DAS algorithm to rank the events and determine which ones to alert on.

This approach would work fine for a batch algorithm that runs the DAS algorithm once a month on the past thirty day's events. However, a spearphishing attack might not be detected by this batch approach until as much as a month after it occurs. Therefore, we now turn to extending our approach to work *in real-time*.

Naively, at the end of each day we could gather the day's events, rank them using DAS, and alert on the β most suspicious. However, this might miss attacks that would have been detected by the batch algorithm, as some days might have many benign but seemingly-suspicious events that mask a true attack.

Instead, we use a more sophisticated algorithm that comes closer to the batch algorithm, yet operates in real time. Each night, our detector collects all the click-in-

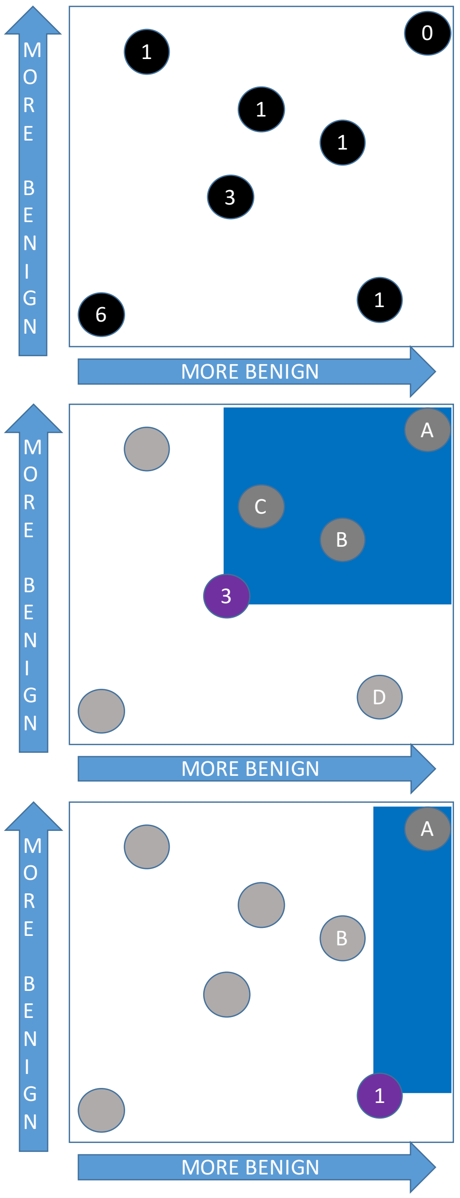


Figure 5: Example diagrams of DAS scores for events in a 2 dimensional feature space. X-values to the right and Y-values toward the top are more benign (thus, values toward the bottom and left are more suspicious). Each circle represents an example event. The number in each circle is the DAS score for the event. For example, looking at the third diagram, the purple event only receives a score of 1. Although the purple event has a more suspicious feature value in the Y dimension than event B, it is more benign in the X dimension. Thus, event B does **not** cause the purple event's score to increment.

email events for the past month and computes their associated feature vectors. For each sub-detector, we rank these events using DAS, select the $30 \times \beta$ most suspicious events, and save them in a set that we call the *ComparisonSet*.

In real time, when our detector observes a click-in-email event from the NIDS, it fetches the event's feature vectors for each impersonation model. Our detector then computes if any of the current click's feature vectors are at least as suspicious as any of the feature vectors in the *ComparisonSet* for its respective impersonation model.[4] If so, our detector generates an alert for the security team. Intuitively, this approach alerts if the event would have been selected by DAS on any day in the past month; or, more precisely, if it is among the 30β most suspicious events in the past month. Our evaluation (§ 6) shows that this real-time approach can safely detect the same attacks as the batch scoring procedure. On some days our real-time approach might generate more alerts than the target budget if a day has a burst of particularly suspicious click-in-email events; however, we show in the next section that this occurs infrequently in practice.

6 Evaluation and Analysis

We evaluated our real-time detector on our dataset of 370 million emails from LBNL, measuring its detection performance (true positives), the time burden (false positives) it imposes on an enterprise's security staff, and how it performs relative to standard anomaly detection techniques that use the same set of features.

For each click-in-email event, we computed its reputation features using log data from a sliding window over the six months prior to the click event. To bootstrap this process, we use the first six months of our dataset as a burn-in period and do not generate alerts for any emails in that period. Later (§ 7), we explore the impact of using a smaller window of historical data to compute feature values.

We configured our detector with a daily budget of 10 alerts per day. LBNL's security team specified 10 alerts per day as a very tolerable number since their team consists of several analysts who routinely process a few hundred alerts each day. To divide this budget among each of our three sub-detectors, we allocated 4 alerts per day for each of the *name spoofer* and *previously unseen attacker* sub-detectors and 2 alerts per day for our *lateral attacker* sub-detector; since lateral spearphishing requires the use of a compromised account, we expect it to occur less often than spoofing-based spearphishing.

6.1 Detection Results: True Positives

Because spearphishing attacks occur infrequently and often go undetected, developing ground truth and measuring true positives is a hard problem. For our evaluation, we draw upon LBNL's incident database, which contains

[4] This is equivalent to running DAS to score the current feature vector against the *ComparisonSet* and checking whether it gives the current feature vector a score of at least 1.

Alert Classification	*Name spoofer*	*Previously unseen attacker*	*Lateral attacker*	Total Count
Spearphish: *known* + successful attack	2	2	2	6 / 7
Spearphish: **unknown** + successful attack	1	1	0	2 / 2
Spearphish: failed attack	3	6	0	9 / 10
Total Spearphish Detected	6	9	2	17 / 19

Table 3: Summary of our real-time detection results for emails in our test window from Sep 1, 2013 - Jan 14, 2017 (1,232 days). Rows represent the type/classification of an alert following analysis by security staff members at LBNL. Columns 2–4 show alerts broken down per attacker model (§ 5.2.2). Column 5 shows the total number of spearphishing campaigns identified by our real-time detector in the numerator and the total number of spearphishing campaigns in the denominator. Out of 19 spearphishing attacks, our detector failed to detect 2 attacks (one that successfully stole an employee's credentials and one that did not); both of these missed attacks fall under the *previously unseen attacker* threat model, where neither the username nor the email address matched an existing entity.

7 known successful spearphishing attacks; this includes 1 spearphishing exercise, designed by an external security firm and conducted independently of our work, that successfully stole employee credentials. Additionally, members of LBNL's security team manually investigated and labeled 15,521 alerts. We generated these alerts from a combination of running (1) an older version of our detector that used manually chosen thresholds instead of the DAS algorithm; and (2) a batched version of our anomaly scoring detector, which ran the full DAS scoring procedure over the click-in-email events in our evaluation window (Sep. 2013 onward) and selected the highest scoring alerts within the cumulative budget for that timeframe.

From this procedure, we identified a total of 19 spearphishing campaigns: 9 which succeeded in stealing an employee's credentials and 10 where the employee clicked on the spearphishing link, but upon arriving at the phishing landing page, did not enter their credentials.[5] We did not augment this dataset with simulated or injected attacks (e.g., from public blogposts) because the true distribution of feature values for spearphishing attacks is unknown. Even for specific public examples, without actual historical log data one can only speculate on what the values of our reputation features should be.

To evaluate our true positive rates, we ran our real-time detector (§ 5.5) on each attack date, with a budget of 10 alerts per day. We then computed whether or not the attack campaign was flagged in a real-time alert generated on those days. Table 3 summarizes our evaluation results. Overall, our real-time detector successfully identifies 17 out of 19 spearphishing campaigns, a 89% true positive rate.

Of these, LBNL's incident database contained 7 known, successful spearphishing campaigns (their incident database catalogues successful attacks, but not ones that fail). Although our detector missed one of these successful attacks, it identified 2 previously undiscovered attacks that successfully stole an employee's credentials. The missed attack used a now-deprecated feature from

[5]A campaign is identified by a unique triplet of ⟨the attack URL, email subject, and email's `From` header⟩.

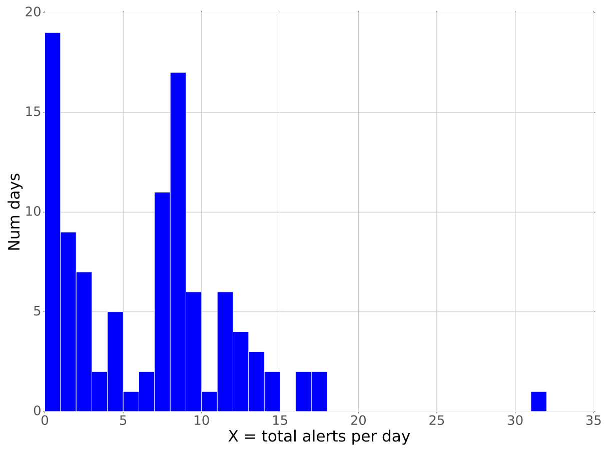


Figure 6: Histogram of the total number of daily alerts generated by our real-time detector (cumulative across all three sub-detectors) on 100 randomly sampled days. The median is 7 alerts/day.

Dropbox [7] that allowed users to host static HTML pages under one of Dropbox's primary hostnames, which is both outside of LBNL's NIDS visibility because of HTTPS and inherits Dropbox's high reputation. This represents a limitation of our detector: if an attacker can successfully host the malicious phishing page on a high-reputation site or outside of the network monitor's visibility, then we will likely fail to detect it. However, Dropbox and many other major file sharing sites (e.g., Google Drive) have dropped these website-hosting features due to a number of security concerns, such as facilitating phishing. Ironically, in the specific case of Dropbox, industry reports mention a large increase in phishing attacks targeted against Dropbox users, where the phishing attack would itself be hosted via Dropbox's website hosting feature, and thus appear to victims under Dropbox's real hostname [11]. Among the attacks that our detector correctly identified, manual analysis by staff members at LBNL indicated that our sub-detectors aptly detected spearphish that fell under each of their respective threat models (outlined in Section 2.1).

6.2 False Positives and Burden of Alerts

At a daily budget of 10 alerts per day, our detector achieved an average false positive rate of 0.004% (the median number of emails per day is 263,086). However, as discussed earlier (§ 5.5), our real-time detector is not guaranteed to produce exactly 10 alerts per day; some days might have a burst of particularly suspicious emails while other days might not have any unusual activity at all. To evaluate the actual daily alert load, we ran our real-time detector on one hundred randomly selected days in our dataset and computed the total number of alerts it generated on each day, shown in Figure 6. From this histogram, we see that while our detector occasionally generates bursts over our target budget, on the vast majority of days (80%) it generates 10 or fewer alerts per day; on nearly 20% of days, it generates no alerts.

During their manual investigation of the 15,521 alerts created during our ground truth generation process, LBNL's security staff tracked how long it took them to investigate these alerts. Surprisingly, LBNL's security staff reported that a single analyst could process an entire month's worth of alerts in under 15 minutes (and thus, on average, under one minute to analyze one day's worth of alerts).

This rapid processing time arises because the analysts were able to develop a two-pass workflow that enabled them to quickly discard over 98% of the alerts, at a rate of 2 seconds per alert; and then follow up with a more in-depth analysis pass (e.g., analyzing detailed HTTP logs and examining the full email headers) over the remaining 2% of alerts, at a rate of 30 seconds per alert. The first pass is so fast because, for the vast majority of our detector's alerts, an analyst could quickly tell if an email constituted a plausible spearphishing threat by inspecting the Subject line, `From` line, and clicked URL of the email. For over 98% of our alerts, this trio of information indicated that the email was highly unlikely to contain a credential spearphishing attack. For example, emails with subjects such as "Never Lose Your Keys, Wallet, or Purse Again!" and "ATTN: Your Stomach Issues FINALLY Explained. See Video Here" are surely not spearphishing attacks.

While the more time-intensive 2% of alerts contained mostly false positives (i.e., not spearphishing), the analysts found two interesting classes of alerts. First, in addition to detecting spearphishing attacks, our detector identified 41 emails from regular phishing campaigns. The analysts distinguished between regular phishing and spearphishing by checking whether the email and HTTP response from the clicked URL contained content that was specifically targeted at LBNL. Second, ironically, our detector generated 40 alerts where the person who clicked on the link in the email was not one of the email's recipients, but rather a member of LBNL's security staff. These clicks were part of routine investigations conducted by LBNL's security staff; for example, in response to a user reporting a suspicious email.

6.3 Anomaly Detection Comparisons

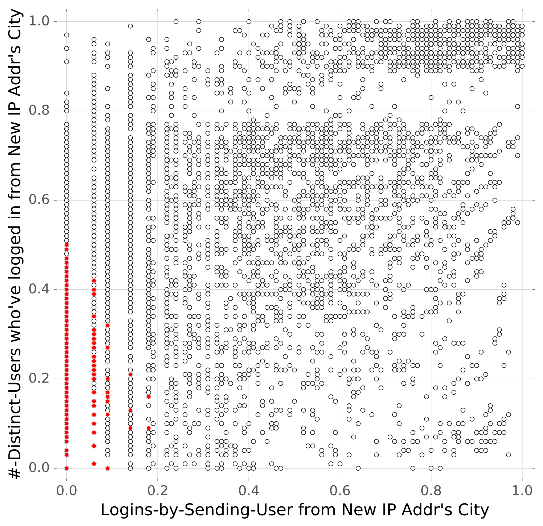
In Section 5.4 we introduced DAS, a simple new technique for anomaly detection on unlabeled data. Now, we evaluate the effectiveness of DAS compared to traditional unsupervised anomaly detection techniques.

We tested three common anomaly detection techniques from the machine learning literature: Kernel Density Estimation (KDE), Gaussian Mixture Models (GMM), and k-Nearest Neighbors (kNN) [5]. To compare the real-time detection performance of each of these classical techniques against DAS's real-time performance, we ran each of these classical techniques using the same training and evaluation procedures we used for our real-time detector's evaluation. Specifically, given the date of each of the 19 attacks and its impersonation model, we extracted the same exact feature values for all click-in-email events that occurred within a thirty day window ending on the attack date; the thirty day window reflected the size of our *ComparisonSet*. We then normalized these feature values and ran each of the three classical anomaly detection techniques on this set of click-in-email events for each attack date. For quantitative comparisons, we computed (1) the number of attacks that would have been detected by each classical technique if it used the same budget that our real-time detector used and (2) the daily budget the classical technique would need to detect all of the attacks that our DAS-driven detector identified.

Like other machine learning methods, these classical algorithms require the user to set various hyperparameters that affect the algorithm's performance. For our evaluation, we tested each classical technique under a range of different hyperparameter values and report the results for whichever values gave the best results (i.e., comparing DAS against the best-case version of these classical techniques).

Table 4 summarizes the results of this comparative experiment. All three traditional techniques detected fewer than 25% of the attacks found by DAS. Moreover, in order for KDE (the best performing classical technique) to detect as many attacks as DAS, it would need a daily budget nearly an order of magnitude larger than ours.

To illustrate why standard unsupervised techniques perform so poorly, the two plots in Figure 7 show the sender reputation features for a random sample of 10,000 *lateral attacker* click-in-email events. The left plot shows the feature values for the actual alerts our DAS detector generated (in red), while the right plot shows the feature values for the alerts selected by KDE using the same budget as our detector. KDE selects a mass



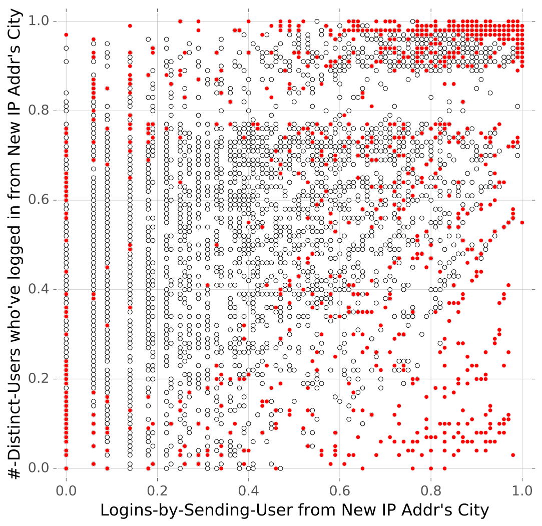


Figure 7: Both plots show the sender reputation feature values (scaled between [0, 1]) of a random sample of 10,000 *lateral attacker* click-in-email events. Filled red points denote events that generated alerts within the daily budget by DAS (left-hand figure) and KDE (right-hand figure).

Algorithm	Detected	Daily Budget
kNN	3/19	10
	17/19	2,455
GMM	4/19	10
	17/19	147
KDE	4/19	10
	17/19	91
DAS (§ 5.4)	17/19	10

Table 4: Comparing classical anomaly detection techniques to our real-time detector, on the same dataset and features. For each of the standard anomaly detection algorithms, the first row shows the number of attacks detected under the same daily budget as ours; the second row shows what the classical technique's budget would need to be to detect all 17 attacks that our real-time detector identified on a daily budget of 10 alerts per day.

of points in the upper-right corner, which illustrates one of limitations of standard techniques discussed in Section 5.4: they do not take into account the directionality of feature values. Because extremely large feature values occur infrequently, KDE ranks those events as highly anomalous, even though they correspond to benign login sessions where the user happened to login from a new IP address in a residential city nearby LBNL. Second, KDE selects a group of events in the bottom-right corner, which correspond to login sessions where an employee logged in from a city that they have frequently authenticated from in the past, but where few other employees have logged in from. KDE's selection of these benign logins illustrates another limitation of standard techniques: they often select events that are anomalous in just one dimension, without taking into account our domain knowledge that an attack will be anomalous in *all* dimensions. Even though the bottom-right corner represents employee logins where few other employees have logged in from the same city, they are not suspicious, because that employee has previously logged in many times from that location: they correspond to benign logins by remote employees who live and work from cities far from LBNL's main campus. Thus, DAS can significantly outperform standard unsupervised anomaly detection techniques because it allows us to incorporate domain knowledge of the features into DAS's decision making.

7 Discussion and Limitations

Detection systems operate in adversarial environments. While we have shown our approach can detect both known and previously undiscovered spearphishing attacks, there are limitations and evasion strategies that adversaries might pursue.

Limited Visibility: Our detection strategy hinges on identifying if an email's recipient engaged in a potentially dangerous action. In the case of credential spearphishing, LBNL's network traffic logs allowed us to infer this behavior. However, our approach has two limitations: first, email and network activity conducted outside of LBNL's network borders will not get recorded in the NIDS logs. Second, LBNL made a conscious decision not to man-in-the-middle traffic served over HTTPS; thus, we will miss attacks where the email links to an HTTPS website. Both of these are typical challenges that network-level monitoring faces in practice. One strategy for alleviating this problem would be to use endpoint monitoring agents on employee machines. Alternatively, a detector could leverage SNI [23] to develop its domain reputation for HTTPS and identify when users visit potentially dangerous HTTPS domains.

In addition to limited network visibility, our detector might miss attacks if a spearphishing email came from a compromised personal email account. Since our detector relies on access to a user's prior login information to detect lateral spearphishing attacks, it will not have the necessary data to compute the features for this sub-detector. To defend against this genre of lateral spearphishing, one could leverage alternative sender reputation features, such as ones based on stylometry [8, 19].

False Negatives and Evasion Strategies: Our detector attempts to meet an upper-bound on the number of alerts it generates. As a result, it might miss some attacks if a number of successful spearphishing campaigns occur on a given day; in effect, the clicks on URLs from the campaigns earlier in the day will mask campaigns that occur later on. To overcome this problem, the security staff could increase the detector's alert budget on days with many attack alerts.

Aside from trying to mask one attack campaign with another, an adversary could attempt to escape detection by crafting an email whose domain or sender reputation features are high. An attacker could boost her link's domain reputation by compromising a frequently visited website and using it to host the credential spearphishing website. This strategy incurs greater costs to execute than modern-day attacks (where an adversary can simply setup her own cheap phishing webpage), and it is unclear whether such an attack would succeed if the site does not normally ask for the employee's corporate credentials. For example, if an adversary compromises a popular video website (e.g., netflix.com), many users might find it unusual for that popular domain to suddenly start asking for the user's enterprise credentials.

Alternatively, an attacker could attempt to inflate the sender reputation features of their adversarial email before using it in an attack. For instance, to prepare a malicious email address for a name spoofing attack, an adversary could start sending emails with the malicious email address and spoofed `From` name for several days before sending a spearphishing email to the targeted recipient. However, the more frequently this address is used, the more the adversary risks someone detecting the adversary's use of a spoofed name; thus this evasion strategy does incur a cost and risk to the attacker.

Future work could explore methods to make DAS more robust. In particular, rather than treating an event E as more suspicious than another event X only if E is more suspicious than X in every dimension, the scoring algorithm could be changed to treat E as more suspicious if it is more suspicious than X in at least *k* dimensions.

Prior History for Feature Extraction: For each click-in-email event, our detector leveraged 6 months of prior log data in order to compute meaningful reputation features. LBNL stores several years worth of logs, so this amount of prior history was easily available for our detector. However, with less historical data, the quality of our detector might degrade (e.g., in the degenerate case with no prior history, all `From` names and addresses will appear as suspicious new entities). To assess how much history our detector needs, we re-ran our evaluation experiments (§ 6.1 and § 6.2) with 3 months of history and with 1 month of history. A 3-month historical window sufficed to detect the same attacks as our 6-month real-time detector, and the median number of alerts per day remained the same (7 per day). However, a detector with only 1 month of history failed to detect one of the attacks and generated a median of 18 alerts per day. With just one month of prior data, too many click-in-email events have the smallest possible feature values; this causes our detector to select entire batches of them because they share the same DAS score.

Extending to Preventative Protection: One could extend our real-time detector to operate in a preventative fashion. As emails arrived, our detector could compute each email's feature values and then check each URL in the email to see whether or not it would generate an alert if the URL were clicked at that moment. If so, we could rewrite the email's URL (before delivering the email to its recipient) to point to an interstitial warning page set up by the enterprise's security team. Our computations show that if we used our real-time detector with a budget of 10 alerts/day, an employee would encounter a median of 2 interstitial pages over the nearly 4-year time span of our evaluation data (Appendix B). Given this low burden, future work could explore how to design effective warning mechanisms as part of a preventative defense.

8 Related Work

Recently, a number of papers have highlighted the threat of spearphishing and explored potential defenses [8, 12, 19, 24]. Closest to our work, the systems proposed by Stringhini et al. [19], Duman et al. [8], and Khonji et al. [12] build behavioral models for senders based on metadata, stylometry, and timing features. They then classify an email as spearphishing or not by using the behavioral model to see whether a new email's features differ from the sender's historical behavioral profile. This prior work cannot detect spearphish sent by a *previously unseen attacker* since the sender has no prior history (and thus no behavioral model to compare the attack email against). More importantly, when they evaluate their systems on smaller datasets with simulated attacks, the best performing detectors obtain false positive rates (FPRs) in the range of 1–10%. Although quite low, an FPR of even 1% is too high for a practical enterprise settings; our dataset contains over 250,000 emails per day, so an FPR of 1% would lead to 2,500 alerts each day. In contrast, our detector can detect real-world attacks, including those from a *previously unseen attacker*, with a budget of 10 alerts per day.

Other work has characterized the landscape of spearphishing attacks against individual activists and dissidents [9, 15, 16, 20]. This body of work shows that targeted attacks on individuals span a wide spectrum of sophistication, from simple third-party tracking software and common exploits to purchasing specialized spyware and exploits from commercial firms. Most recently, LeBlond et al. conducted a large-scale analysis of exploit documents used in targeted attacks [14]. Their analysis found that none of these malicious attachments used a zero-day exploit, and over 95% of these documents relied on a vulnerability that was at least one year old. While these attacks can succeed against vulnerable activists and individuals, such dated exploits will likely fail against an enterprise with good security hygiene. Indeed, over the past few years, all of the spearphishing attacks on LBNL have been credential spearphishing.

9 Conclusion

In this work, we developed a real-time detector for identifying credential spearphishing attacks in enterprise settings. Two key contributions enabled our detector to achieve practical performance: (1) a new set of features that targets the two fundamental stages of successful spearphishing attacks, and (2) a new anomaly detection technique that leverages these features to detect attacks, without the need for any labeled training data.

We evaluated our approach on an anonymized dataset of over 370 million emails collected from a large national lab. At a false positive rate of less than 0.005%, our system detected all but two attacks in our dataset and uncovered two previously unknown successful attacks. Compared against our anomaly scoring technique, standard anomaly detection techniques would need to generate orders of magnitude more false positives to detect the same attacks as our algorithm. Because of our approach's ability to detect a wide range of attacks, including previously undiscovered attacks, and its low false positive cost, LBNL has implemented and deployed a version of our detector.

Acknowledgements

The authors would like to thank Nicholas Carlini, Chris Thompson, Alyosha Efros, and Paul Pearce for insightful discussions and feedback. We would also like to thank Partha Banerjee, Jay Krous, Stephen Lau, Vincent Stoffer, and Adam Stone of the Lawrence Berkeley National Laboratory for their wide-ranging assistance in conducting the research. This work was supported by the AFOSR under MURI award FA9550-12-1-0040; the National Science Foundation through CNS awards 1161799/1237265/1348077/1406041, Intel through the ISTC for Secure Computing, Qualcomm, Cisco, the Hewlett Foundation through the Center for Long-Term Cybersecurity, an NSF Graduate Fellowship, and a Facebook Fellowship.

References

[1] Spear phishing: The top ten worst cyber attacks. `https://blog.cloudmark.com/wp-content/uploads/2016/01/cloudmark_top_ten_infographic.png`.

[2] DMARC. `https://dmarc.org/`, 2016.

[3] Peter Bright. Spearphishing + zero-day: RSA hack not "extremely sophisticated". `http://arstechnica.com/security/2011/04/spearphishing-0-day-rsa-hack-not-extremely-sophisticated/`, April 2011.

[4] Elie Bursztein and Vijay Eranti. Internet-wide efforts to fight email phishing are working. `https://security.googleblog.com/2013/12/internet-wide-efforts-to-fight-email.html`, Feb 2016.

[5] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, 2009.

[6] Nitesh Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter*, 6(1):1–6, 2004.

[7] Dropbox Community. Discontinuing rendering of html content. `https://www.dropboxforum.com/t5/Manage-account/Discontinuing-rendering-of-HTML-content/td-p/187920`, Sep 2016.

[8] Sevtap Duman, Kubra Kalkan-Cakmakci, Manuel Egele, William Robertson, and Engin Kirda. Emailprofiler: Spearphishing filtering with header and stylometric features of emails. In *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, pages 408–416. IEEE, 2016.

[9] Seth Hardy, Masashi Crete-Nishihata, Katharine Kleemola, Adam Senft, Byron Sonne, Greg Wiseman, Phillipa Gill, and Ronald J. Deibert. Targeted threat index: Characterizing and quantifying politically-motivated targeted malware. In *Proceedings of the 23rd USENIX Conference on Security Symposium*, SEC'14, 2014.

[10] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[11] Nick Johnston. Dropbox users targeted by phishing scam hosted on dropbox. `https://www.symantec.com/connect/blogs/dropbox-users-targeted-phishing-scam-hosted-dropbox`, Oct 2014.

[12] Mahmoud Khonji, Youssef Iraqi, and Andrew Jones. Mitigation of spear phishing attacks: A content-based authorship identification framework. In *Internet Technology and Secured Transactions (ICITST), 2011 International Conference for*, pages 416–421. IEEE, 2011.

[13] Aleksandar Lazarevic, Levent Ertoz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 25–36. SIAM, 2003.

[14] Stevens Le Blond, Cédric Gilbert, Utkarsh Upadhyay, Manuel Gomez Rodriguez, and David Choffnes. A broad view of the ecosystem of socially engineered exploit documents. In *NDSS*, 2017.

[15] Stevens Le Blond, Adina Uritesc, Cédric Gilbert, Zheng Leong Chua, Prateek Saxena, and Engin Kirda. A look at targeted attacks through the lense of an ngo. In *USENIX Security*, pages 543–558, 2014.

[16] William R Marczak, John Scott-Railton, Morgan Marquis-Boire, and Vern Paxson. When governments hack opponents: A look at actors and technology. In *USENIX Security*, pages 511–525, 2014.

[17] Trend Micro. Spear-phishing email: Most favored APT attack bait. *Trend Micro, http://www.trendmicro.com.au/cloud-content/us/pdfs/security-intelligence/white-papers/wp-spear-phishing-email-most-favored-apt-attack-bait.pdf (accessed 1 October 2014)*, 2012.

[18] Steve Sheng, Mandy Holbrook, Ponnurangam Kumaraguru, Lorrie Faith Cranor, and Julie Downs. Who falls for phish?: a demographic analysis of phishing susceptibility and effectiveness of interventions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 373–382. ACM, 2010.

[19] Gianluca Stringhini and Olivier Thonnard. That ain't you: Blocking spearphishing through behavioral modelling. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 78–97. Springer, 2015.

[20] Colin Tankard. Advanced persistent threats and how to monitor and deter them. *Network security*, 2011.

[21] Lisa Vaas. How hackers broke into John Podesta, DNC Gmail accounts. `https://nakedsecurity.sophos.com/2016/10/25/how-hackers-broke-into-john-podesta-dnc-gmail-accounts/`, October 2016.

[22] Colin Whittaker, Brian Ryner, and Marria Nazif. Large-scale automatic classification of phishing pages. In *NDSS*, volume 10, 2010.

[23] Wikipedia. Server name indication. `https://en.wikipedia.org/wiki/Server_Name_Indication`, June 2017.

[24] Mengchen Zhao, Bo An, and Christopher Kiekintveld. Optimizing personalized email filtering thresholds to mitigate sequential spear phishing attacks. In *AAAI*, pages 658–665, 2016.

A Feature Vectors and Comparators per Sub-Detector

Name spoofer Features	Comparator for DAS
Host age of clicked URL (email ts − domain's 1st visit ts)	$\leq$
# visits to clicked URL's host prior to email ts	$\leq$
# weeks that `From` name has sent email on ≥ 5 days	$\geq$
# days that `From` name and `From` addr have appeared together in emails	$\leq$

Table 5: Summary of the feature vector for our *name spoofer* sub-detector and the "suspiciousness" comparator we provide to DAS for each feature.

Previously unseen attacker Features	Comparator for DAS
Host age of clicked URL (email ts − domain's 1st visit ts)	$\leq$
# visits to clicked URL's host prior to email ts	$\leq$
# days that `From` name has sent email	$\leq$
# days that `From` addr has sent email	$\leq$

Table 6: Summary of the feature vector for our *previously unseen attacker* sub-detector and the "suspiciousness" comparator we provide to DAS for each feature.

Lateral attacker Features	Comparator for DAS
Host age of clicked URL (email ts − domain's 1st visit ts)	$\leq$
# visits to clicked URL's host prior to email ts	$\leq$
# distinct employees who have previously logged in from the same city as the session's new IP addr	$\leq$
# previous logins by the current employee from the same city as the session's new IP addr	$\leq$

Table 7: Summary of the feature vector for our *lateral attacker* sub-detector and the "suspiciousness" comparator we provide to DAS for each feature.

B Preventative Interstitials

In Section 7 we discussed how to extend our detector from a realtime alert system to a preventative defense by rewriting suspicious URLs in emails to redirect to an interstitial page. This defense can only be practical if it does not cause employees to frequently land on interstial'ed pages. To assess this concern, we ran our detector on our entire evaluation dataset (Sep 1, 2013 – Jan 14, 2017) with an average daily budget of 10 alerts, and selected the alerts that fell within our cumulative budget for that window (i.e., selecting the top $B = 10 * N_{daysInEvalWindow} = 12,310$ most suspicious click-in-email events). For each recipient (`RCPT TO` email address) that received the emails of those 12,310 alerts, we computed the number alerts that recipient received over the entire evaluation time window. Figures 8 and 9 show these results in histogram and CDF form.

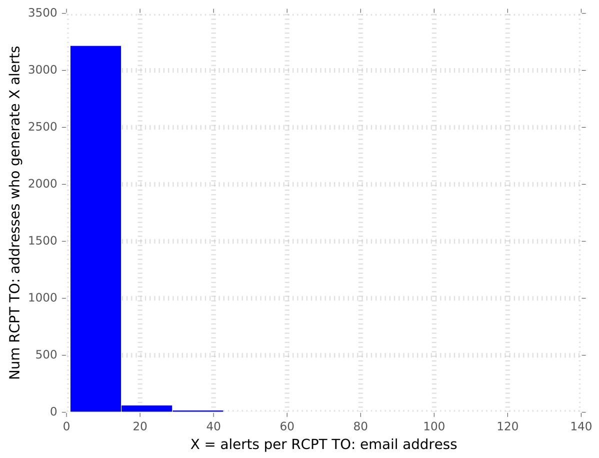


Figure 8: Histogram of alerts per `RCPT TO` address for our detector using an average budget of 10 alerts per day across the Sep 1, 2013 – Jan 14, 2017 timeframe.

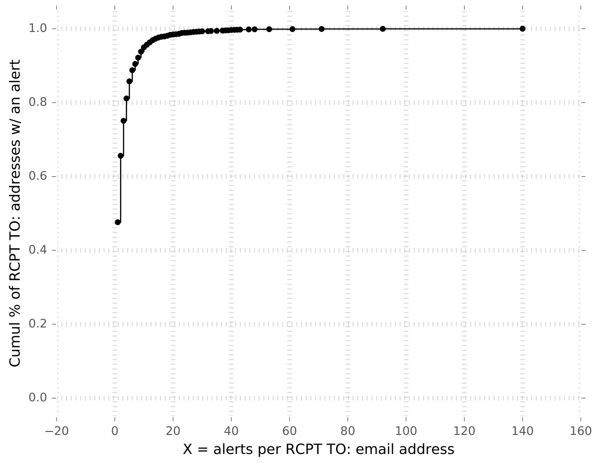


Figure 9: CDF of alerts per `RCPT TO` address for our detector using an average budget of 10 alerts per day across the Sep 1, 2013 – Jan 14, 2017 timeframe.

From these figures, we see that over 95% of employees would see fewer than 10 interstitials across the entire time span of nearly 3.5 years.