Reading Between The Lines: Content-Agnostic Detection of Spear-Phishing Emails

Hugo Gascon[1], Steffen Ullrich[2], Benjamin Stritter[3], and Konrad Rieck[1]

[1] TU Braunschweig
[2] Genua GmbH
[3] Friedrich Alexander-Universität Erlangen-Nürnberg

Abstract. Spear-phishing is an effective attack vector for infiltrating companies and organisations. Based on the multitude of personal information available online, an attacker can craft seemingly legit emails and trick his victims into opening malicious attachments and links. Although anti-spoofing techniques exist, their adoption is still limited and alternative protection approaches are needed. In this paper, we show that a sender leaves content-agnostic traits in the structure of an email. Based on these traits, we develop a method capable of learning profiles for a large set of senders and identifying spoofed emails as deviations thereof. We evaluate our approach on over 700,000 emails from 16,000 senders and demonstrate that it can discriminate thousands of senders, identifying spoofed emails with 90% detection rate and less than 1 false positive in 10,000 emails. Moreover, we show that individual traits are hard to guess and spoofing only succeeds if entire emails of the sender are available to the attacker.

Keywords: Spear-phishing · Email Spoofing · Targeted attack detection

1 Introduction

Emails are a prevalent attack vector for infiltrating companies and organisations. As documents and links are regularly exchanged via email within and across these environments, they are a perfect vehicle for transmitting malicious payloads to a victim [6, 20]. To increase their success, attackers specifically target individual members of an organization using carefully crafted emails—a technique referred to as *spear-phishing*. For example, an attacker may pick an appropriate topic, use correct wording and spoof a well-known sender to convince the recipient of the veracity of an email [16]. These targeted attacks are more advanced than regular phishing or spam campaigns, as they are individually adapted to the environment and behavior of the victim. As a result, there exist only few similarities between different spear-phishing attacks which makes it hard to construct effective defenses.

Although users are increasingly aware of the risk they are exposed to, they have to rely on hints provided by the email client to detect spoofed content. In the default setup, several clients, like Microsoft Outlook and Mozilla Thunderbird, display only little information for identifying the sender, such as the `From` and `Reply-To` fields. Emails from unknown senders can be marked accordingly and specifically dealt with but these and other fields can be forged, making it hard even for a skilled user to distinguish legitimate content from well-crafted attacks [5, 34]. While inconsistent combinations of these fields can be easily detected and used to notify the user of a potential threat, the situation becomes challenging if all fields are correctly adapted by the adversary, such that the email appears totally legitimate in its content as well as its headers.

Common anti-spoofing techniques such as the Sender Policy Framework [SPF, 24], DomainKeys Identified Mail [DKIM, 7] and the more recent Domain Message Authentication Reporting & Conformance [DMARC, 25] can help to validate the sender of an email in this situation. Similarly, techniques for digital signing of emails, such as PGP [4] and S/MIME [29], enable to verify the sender. Unfortunately, these techniques are still not widely adopted in practice. While we notice several email domains in our evaluation data with SPF entries, less than 5% of the collected 700.000 emails contain corresponding DKIM headers or even digital signatures. Moreover, all of these techniques need to be implemented at the sending side, which renders it difficult to protect from spoofing if not all communication parties adopt the technology [13, 28]. Therefore, given an attacker that is able to exactly match the address of a known sender, the user is unable to detect the attack and might be tricked into opening a malicious file or link.

As a result, there is a demand for alternative approaches to protect users from highly targeted spear-phishing emails. In this paper, we propose a method that is able to verify, without relying on its content, if an email exactly matching the address of a known sender truly originates from its legit source. Our method builds on the observation that a sender leaves characteristic traits in the structure of an email, which are independent from textual content and often persist over time. These traits significantly differ between senders and reflect peculiarities of the user behavior, email client and delivery path, such as particular header combinations, encoding formats and attachment types. Based on this observation, we develop a detection method that receives the mailbox of a user as input and applies machine learning techniques to generate profiles for all senders in the mailbox, even if only a few emails are available. These profiles provide a content-agnostic view on the sender and enable us to spot spoofed emails as deviations from the learned profiles.

We empirically evaluate our approach on a collection of 92 mailboxes from twelve different domains, covering over 700,000 emails from 16,000 senders. We demonstrate that our method can discriminate thousands of senders in one mailbox and enables identifying spoofed emails with 90% detection rate and less than 1 false positive in 10,000 emails. Moreover, we can show that the individual traits of a sender observed at the recipient's end are hard to guess and spoofing attempts only succeed if entire emails of the sender as delivered to the recipient are known to the adversary. Although our approach cannot generally rule out spoofing due to leaked emails, it considerably raises the bar for targeted attacks

```
Return-Path: <john@doe.com>
Received: from [93.184.216.34] (HELO example.com)
    by example.com with ESMTP id 69815728;
    Tue, 16 May 2017 14:06:48 +0200
To: Jane Dee <jane@example.com>
Date: Tue, 16 May 2017 14:00:02 +0200
Message-Id: <20170516133920.23212@doe.com>
Subject: Security Conference
From: John Doe <john@doe.com>
In-Reply-To: <1405590537$56fe@example.com>
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="boundary"

--boundary
Content-Type: text/plain

For your interest: https://tinyurl.com/y9tfoet7

--boundary
Content-Type: application/octet-stream; name="x.exe"
Content-Transfer-Encoding: base64

TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAKCkdyZWV0
aW5ncywgUmV2aWV3ZXIhCsKvXF8o44OEKV8vwq8KCg==
--boundary--
```

Fig. 1: Simplified email as running example.

and—in absence of widely deployed server-side solutions—provides an effective protection for companies and organisations targeted by spear-phishing attacks.

In summary, we make the following contributions:

– *Characteristic sender profiles:* We identify traits which enable us to characterize the sender of an email without relying on textual content. The resulting profiles are expressive enough to distinguish thousands of senders while accounting for the diversity of individual emails.

– *Detection of spear-phishing emails:* We demonstrate how the learned profiles of senders can be used for identifying spoofed emails and help to mitigate the risk of spear-phishing attacks in absence of stronger server-side solutions in practice.

– *Evaluation and evasion experiments:* We evaluate the performance of our method through a series of increasingly adverse scenarios where the attacker becomes stronger by obtaining more information about the target and building a better model of the spoofed sender.

The rest of this paper is organized as follows: In Section 2 we present traits observable in the structure of emails and describe in Section 3 how these can be used to construct profiles for senders. We evaluate the resulting detection method in Section 4 and discuss its impact and limitations in Section 5. Related work is reviewed in Section 6 and Section 7 concludes the paper.

2 Traits in Email Structure

The identification of spoofed emails is a challenging problem of network security. An attacker can almost arbitrarily manipulate the structure and content of emails, ranging from a trivially spoofed `From` field to carefully crafted sequences of fake `Received` headers [see 30]. In absence of exact detection techniques in practice, such as DKIM and DMARC, it is thus hard to discriminate legitimate from forged emails.

The freedom available for constructing a spoofed email, however, may also turn against the attacker and pose an obstacle. We argue that it is non-trivial to mimic an email from a particular sender without detailed knowledge and that minor irregularities in the email structure may provide valuable clues for identifying spear-phishing attacks. If the attacker has access to emails from a sender known to the victim, she can simply copy the email structure, yet if this information is not fully available, she needs to make a good guess and hope that the forged structure mimics the original communication well.

For uncovering such forgeries, we identify three groups of traits that can characterize the sender of an email: First, when writing an email the sender introduces *behavior features* that reflect individual preferences and peculiarities. Second, the email client generates *composition features*, identifying the particular client and its configuration. Third, the delivery of an email leaves *transport features* that capture details of the sending and receiving infrastructure. In the following, we describe these groups of traits in more detail and use the simplified email in Figure 1 as a running example through out this section.

2.1 Behavior Features

When a user writes an email, several of her individual preferences can manifest in the structure of the email—aside from her writing style and habits [10, 33]. For example, some senders are frequently including recipients using the `CC` header, whereas others avoid this and prefer to address all recipients directly using the `To` field. Similarly, senders differ in the type and amount of files they are attaching to emails in conversations. While some of these features are volatile and change between different contexts, other features may persist over time and provide a first basis for constructing a profile of the sender.

For our analysis, we identify 13 feature types that characterize the behavior of a sender in the structure of an email, including

1. the type, number and order of attachments, for example when multiple documents are exchanged,
2. the relation to other emails and recipients, for example in form of `References` and `In-Reply-To` headers,
3. digital signatures and certificates attached to the email as well as corresponding PGP and S/MIME fields, and
4. the amount of text in the main part and the amount of quoted text in email responses.

A complete list of all 13 features is provided in Table 4 of the appendix. Note that the cardinality of these features differs, where some may appear multiple times in an email, such as the type of attachments and others only once, such as the depth of the MIME structure. As an example, the email in Figure 1 shows the attachment of an executable file (line 20) and the reference to a previous conversation (line 10)—two features that are rarely used in combination.

2.2 Composition Features

The second source for traits in the structure of an email is the mail user agent (email client) that converts the provided addresses, text and attachments into a suitable format for delivery. As emails have been originally restricted to ASCII characters, there exists a wealth of encoding schemes for converting binary data to a compatible ASCII representation [e.g., 14, 15, 23]. These schemes are selected by the email client and often slightly vary in implementation, thus providing features that characterize the composition of an email. For example, the Base64 encoding [23] does not enforce a fixed text length and thus clients differ in the formatting of the corresponding text blocks. Similarly, there exists several minor variations in the construction of multi-part MIME messages that provide clues about the client and its configuration.

For our analysis, we identify 22 feature types that capture peculiarities of the email client and its configurations, including

1. the type, order and syntax of common headers, such as the `From`, `To`, `Subject` and `Message-Id` headers,
2. the type, order and syntax of headers in MIME parts, including fields like `Content-Type` and `Content-Disposition`,
3. the syntax of address fields, such as the formatting and quoting of names and email addresses,
4. the encoding of international characters in the subject field, in address fields and filenames,
5. the type and location of textual content, such as HTML and plain parts in the email,
6. client-specific behavior, such as the length of line breaks, missing and superfluous encodings of characters,
7. individual details of the MIME structure, such as the depth and the order of different MIME parts, and
8. the structure of the `Message-Id` header and the structure of MIME boundaries.

A complete list of the 22 composition features is provided in Table 5 of the appendix. While these features alone are clearly not sufficient to identify attacks, in combination with behavior and transport features they sharpen the view on a sender and thereby obstruct the spoofing of email addresses. As an example, the email in Figure 1 shows a unique order of the `From`, `To` and `Subject` field (line 5–9) which indicates a rare email client. Furthermore, the Base64-encoded attachment is formatted using a 60 character line length (line 23).

2.3 Transport Features

A third group of traits can be attributed to the delivery path of an email. As the email moves from the sending to the receiving mail transport agent, often passing multiple hops, different headers are added to the structure. These headers describe the individual mail hops in form of `Received` headers and provide information about available delivery features, such as delivery protocols, TLS or the time zone of the mail server. These headers and features, again, generate a series of traits that can help to distinguish different senders and spot irregularities in the delivery process.

Although an attacker can insert fake headers prior to the delivery of an email, it is not possible to change or remove headers added by hops on the delivery path. As a consequence, an attacker can only forge these headers by either connecting directly to the receiving server or, alternatively, attempting to inject emails early into the delivery process—a tractable but non-trivial task in practice, as it would require having access to the same delivery infrastructure as the sender that the attacker is trying to spoof.

We identify 11 transport features that enable us to reconstruct the delivery path of an email and spot deviations from past emails of the same sender. These features include

1. the number and order of `Received` headers, where each hop is represented by the hash of its hostname,
2. the path of time zone from the first to the last hop during the delivery process,
3. the delivery protocols and TLS features available in some `Received` headers,
4. the validity of DKIM records added by the servers and their relation to the claimed sender of the email, and
5. non-standard headers added by spam filters or anti-virus services during the delivery of the email.

Table 6 in the appendix provides a list of all 11 transport features. As an example of traits introduced by the delivery process, the email in Figure 1 contains a detailed `Received` header (line 2–4). This header defines the mail hop, delivery protocol and delivery time. This information is available with any mail passing the hop and thus can leak to the attacker. However, we show in Section 4 that knowledge of transport features alone is insufficient to evade our detection method and that the attacker needs access to original emails delivered to the recipient for successfully spoofing a sender.

3 Detection Methodology

Equipped with three groups of traits for characterizing the sender of an email, we are ready to develop a corresponding detection method using machine learning techniques. The application of learning methods spares us from manually constructing detection rules for each of the senders and thereby allows for scaling our approach to thousands of senders, as we demonstrate in Section 4.

3.1 Feature Extraction and Embedding

The proposed groups of traits provide detailed information about the structure of emails from each sender in the recipient's mailbox. In order to learn a profile from the traits, however, we require a numerical representation that can be used in combination with common learning methods. As a remedy, we apply the concept of a *bag-of-words model*—a technique originating from information retrieval [32] and natural language processing [21, 22]—and adapt it to the traits extracted from the structure of emails.

To this end, we represent each of the extracted traits as a feature string and build a joint set F that comprises all observable strings from the three groups of traits:

$$F := F_{\text{behavior}} \cup F_{\text{composition}} \cup F_{\text{transport}}.$$

Making use of this set F, we define an $|F|$-dimensional vector space that takes values 0 or 1 in each dimension. Each email e is then mapped to this space by building a vector $\varphi(e)$, such that for each feature f extracted from e the corresponding dimension is set to 1, while all other dimensions are set to 0. Formally, this map can be defined for all emails M as

$$\varphi : M \longrightarrow \mathbb{R}^{|F|}, \quad \varphi(e) \longmapsto \left(I_f(e)\right)_{f \in F}$$

where the auxiliary function I simply indicates whether the feature f is present in e, that is,

$$I_f(e) = \begin{cases} 1 & \text{if email } e \text{ contains feature } f \\ 0 & \text{otherwise.} \end{cases}$$

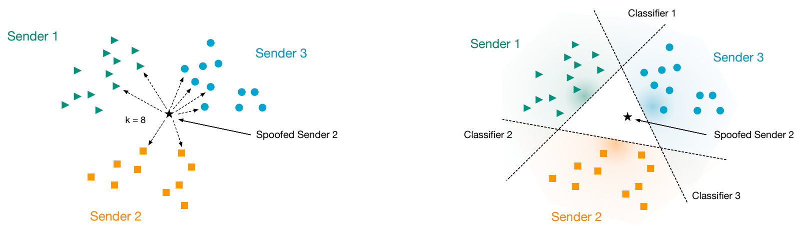
The resulting binary vector space $\mathbb{R}^{|F|}$ allows us to represent each email as a vector of the contained traits of its sender. In the following, we describe how we use this representation to train a machine learning classifier that, based on these features, is able to assign each email to its corresponding sender and indicate possibly spoofed emails.

3.2 Model Learning and Classification

Several learning methods can be applied for classifying data in a vector space. To operate in our setting, however, a learning method needs to address additional requirements: First, the method has to be able to operate in a high-dimensional vector space, as the set F may cover thousands of different traits. Second, the method needs to be capable of learning a classification model, even if only very few training data is available, such as a couple of emails only.

In view of these requirements, we select the following two learning methods for our detection approach: (a) a k-nearest-neighbors classifier (kNN) that can generate good classification results with very few training data and (b) a multi-class support vector machine (SVM) which is known for effectively operating in high-dimensional vector spaces [see 9].

K-Nearest Neighbors Classifier The kNN algorithm is a simple yet effective learning method for classification. It computes the distance between a test sample and all existing samples in a training set and makes a decision through voting on



(a) kNN classifier **(b)** SVM one-vs-all classifier

Fig. 2: Schematic overview of the detection: A classifier is used to identify emails as spoofed when a mismatch between the output of the classifier and the origin sender address occurs.

the labels of its k-nearest samples after applying a weight function (see Figure 2). Such instance-based learning algorithms do not construct an explicit learning model and thus can be applied even if only a single email is available for a sender. For our approach, we label each feature vector with the address of the originating sender address. When a new email is received, we compute the distance between this sample and the feature vectors of all existing emails as follows

$$d(e_x, e_y) = \big|\big|\, \varphi(e_x) - \varphi(e_y) \,\big|\big|_1 = \sum_{f \in F} \big|\, I_f(e_x) - I_f(e_y) \,\big|\,,$$

where d corresponds to the Manhattan or L_1 distance. A mismatch between the incoming sender address and the prediction of the classifier is then flagged by our method as a spoofing attempt.

The advantage of making predictions with very few training data, however, comes at a price. The distance between each new email and all existing emails needs to be computed before making a decision, which is computationally expensive on large mailboxes. Fortunately, this problem can be addressed in two ways: First, one can implement the classifier using special data structures for reducing the number of distance computations, such as ball trees and cover trees [2]. Second, if the number of training instances reaches a certain limit, one can simply switch to another learning method, such as a support vector machine or, when possible, sample the training data according to a distribution that maintains the classifier performance.

Multi-Class Support Vector Machines As second learning method, we employ a linear multi-class SVM algorithm [11]. The algorithm computes a series of maximum-margin hyperplanes that separate the emails from one sender from the emails of all other senders (see Figure 2b). That is, given N different senders, N hyperplanes are determined, each one of them represented by a vector $w \in \mathbb{R}^{|F|}$ and a scalar b in the vector space.

If a new email arrives, we simply determine the position to the learned hyperplanes and pick the sender with the best match, that is, the largest value of

$$h(e) = \langle \varphi(e), w \rangle + b = \sum_{f \in F} I_f(e) \cdot w_f + b.$$

Note that this function can be computed efficiently, if the feature vector $\varphi(e)$ is sparse, as only non-zero dimensions $I_f(e)$ contribute to the output. As a result, we can compute $h(e)$ in linear time in the number of traits $|e|$ extracted from e and the overall run-time for analyzing an email is $O(N|e|)$. In contrast to the kNN algorithm, the run-time for the prediction of a linear SVM is independent of the size of the training set and thus this learning method is suitable if more emails are available from particular senders [see 11].

4 Evaluation

We proceed to evaluate our detection method on a large dataset of real-world emails. In particular, we are interested in studying the ability of our method to characterize the sender of an email based on its structure and to identify spoofed emails under different levels of knowledge of the adversary. Before presenting these experiments, we first introduce our dataset (Section 4.1) and define the corresponding attacker model (Section 4.2).

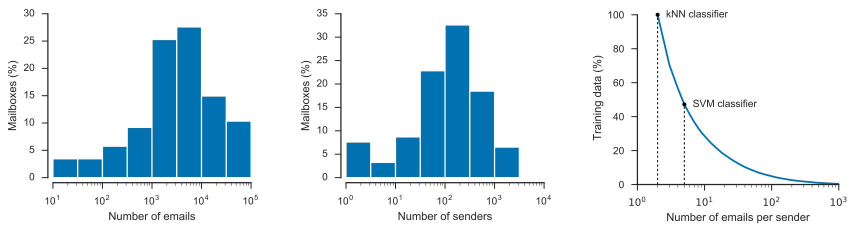
4.1 Evaluation Data

For our evaluation, we have gathered anonymized features extracted from 92 mailboxes from twelve different domains, including enterprise and commercial email services. To evaluate the efficacy of our detection method, we require at least one email for learning and one for testing from each sender. Consequently, we discard all emails from senders that have sent only a single email. Our final dataset comprises a total of 760,603 emails from 17,381 senders, where each sender has authored at least two emails. These emails are described by a total of 617,960 features extracted using the traits defined in Section 2. Table 1 provides an overview of the statistics of our evaluation data.

Table 1: Statistics of evaluation data.

Basic statistics			**Total**
Mailboxes			92
Emails			760,603
Senders			17,381
Features			617,960
Detailed statistics	**Min.**	**Mean**	**Max.**
Emails per mailbox	2	8,267	50,924
Emails per sender	2	43	44,204
Senders per mailbox	1	279	2,144
Features per email	5	69	183
Emails per sender and mailbox	2	29	10,304

Figure 3 depicts in more detail how emails and senders are distributed within our dataset. From Figure 3a and 3b we can see that over 50% of the mailboxes in our dataset contain between 10^3 to 10^4 emails and between 10^2 to 10^3 different senders. This large corpus of emails provides a good basis for evaluating the performance of our method. Depending on the applied learning model, however, we require a minimum number of emails per sender and thus not all senders might be available for training. Figure 3c shows the amount of training data available to a learning method depending on the minimum number of emails per sender. While for the kNN classifier all senders can be used for evaluation, in the case of the SVM classifier, we need to restrict our experiments to 46% of the data, as we require at least 5 emails for training.



(a) (b) (c)

Fig. 3: Overview of the evaluation data: (a) distribution of emails and (b) distribution of senders in the 92 mailboxes; (c) training data available for learning with varying emails per sender.

To prepare our experiments, we extract feature vectors from all emails in our evaluation data. This may seem as an intractable task at first glance, as the resulting vector space has over 600,000 dimensions. However, the majority of these dimensions is zero and each email contains only between 5 to 183 features (see Table 1). As a result, we can make use of efficient data structures for operating with these sparse feature vectors [see 31].

As a sanity check whether our representation is suitable for learning a classification, we first study how senders in a mailbox differ from each other and then analyze how emails from a specific sender change over time. To this end, we first calculate a simple statistic: For each sender, we compute the average of its feature vectors and measure the distances between the resulting 17,381 mean vectors within each mailbox. We make use of the Manhattan distance (L_1 distance) for comparing the mean vectors. The distance can be interpreted as the average number of features differing between the senders and thus provides an estimate for the quality of extracted traits.

Figure 4 shows the distribution of the Manhattan distances between all senders in each mailbox. It can be observed that most senders are separated from each other by a distance larger than 40 on average. This demonstrates that several of the extracted traits are highly specific and capture nuances of the email structure suitable for discriminating the senders. Multiple sources may introduce variability and noise into the email traits of a sender, such as software updates, network configurations and changing devices. We thus study how emails from an individual sender change over time. In particular, we want to answer the question how many features change in a new email when it is compared with existing emails from the same sender. For this, we measure the Manhattan distance between each email received at a certain point in time in a mailbox and all emails previously received from the same sender. The average number of differing features is then presented as a percentage of the feature space dimensionality. Figure 5 shows that a slight feature drift exits. It can be observed how the variability grows rapidly at first with the initial emails received from a sender. However, when an increasing number of emails is received each class becomes more compact and the average percentage of different features in a new email decreases. Note that although profiles become more stable during time, they also tend to differ considerably as shown in Figure 4.

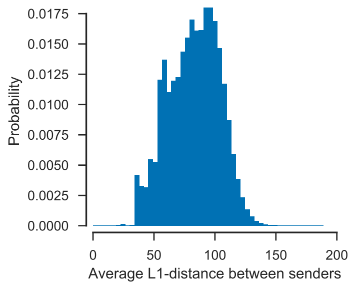


Fig. 4: Distance between senders

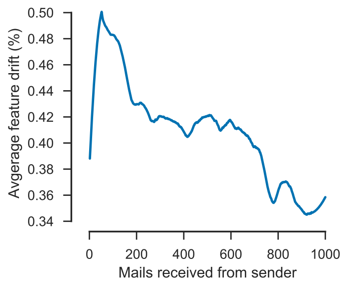


Fig. 5: Feature drift over time

As the final preparation step, we determine the presence of anti-spoofing techniques in the 760,603 emails using corresponding email client and transport features. Table 2 shows the percentage of emails in our dataset that contain anti-spoofing techniques, where we additionally report statistics from the top million web domains listed at the monitoring service *BuiltWith* [3]. Although the adoption of SPF [24] has reached almost 40% by now, the overall implementation of anti-spoofing techniques is still low in both data sources. In particular, recent techniques, such as DKIM [7] and DMARC [25] are used in less than 5% of the emails, thereby emphasizing the need for alternative protection measures.

4.2 Attacker Model

Table 2: Anti-spoofing techniques in our evaluation data and as reported by the monitoring service *BuiltWith*.

Anti-spoofing technique	**Our data**	**Top 1M [3]**
SPF	—	39.9%
DKIM	4.3%	0.1%
DMARC	—	1.3%
PGP, S/MIME	0.88%	—

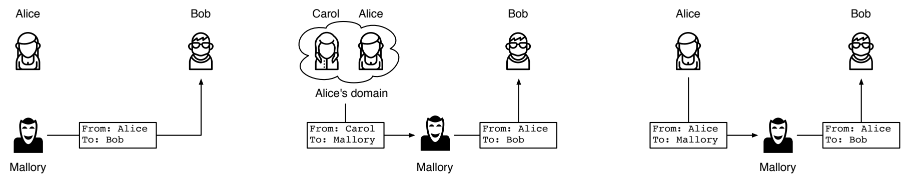
In the absence of anti-spoofing techniques, a skilled adversary is able to forge most of the data contained in an email. However, we argue that, by inferring a sender profile based on traits of the email structure, an attacker is forced to mimic such profile to effectively masquerade as the sender. As a consequence, the success of such spoofing depends on how much information of the email structure is available to the adversary and if the attacker has access to the senders delivery infrastructure.

Therefore, we begin the evaluation of our approach by measuring in a controlled experiment how an attacker may affect the detection performance by spoofing an increasing number of features from a sender's profile (i.e. all features extracted from all emails received from a specific sender in a mailbox). To this end, we first split each sender's data in a mailbox into training and testing sets and then train both kNN and SVM classifiers. For testing, we select random emails from other mailboxes and relabel them as known senders of the target mailbox to imitate spoofing attempts. This means that our testing set is comprised of 50% of legitimate emails and 50% of spoofed emails with a random percentage of correct traits of the target sender.

Note that to generate spoofed emails we do not rely on their textual content for feature extraction. Moreover, we adapt the transport features added by the recipient MTA to the recipient mailbox. As a result, the spoofed emails in our testing set are not different from real spear-phishing emails sent by an attacker, as no textual content is considered.

We measure the detection performance of our classifiers using the true-positive rate (TPR) and false-positive rate (FPR). In our setup, a true positive implies that a spoofed email has been correctly identified, while a false positive corresponds to a legitimate email wrongly being tagged as spoofed. Furthermore, we use a Receiver Operating Characteristic (ROC) curve to present both evaluation metrics and calculate the area under the ROC curve (AUC) as a numerical aggregate of the classification performance [see 12]. Although an adversary with increasing capacity will affect the ability of the classifier to correctly identify deviations from a user profile, the information available to an attacker is constrained by threat scenarios that can occur in reality. In this work, we thus assume that the knowledge of an attacker can range from knowing nothing about the spoofed sender to having real examples of her emails.

Accordingly, we model these attackers through a series of increasing adversarial setups and proceed to evaluate the performance of our approach in each scenario as depicted in Figure 6:



(a) Blind Spoofing **(b)** Known Domain **(c)** Known Sender

Fig. 6: Threat scenarios for increasing attacker capabilities based on the acquired knowledge about the spoofed sender: (a) the attacker has no information about the sender, (b) the attacker has access to emails received from the sender's domain and, (c) the attacker has access to one or more emails from the real sender.

(*a*) *Blind Spoofing:* In this scenario, the attacker (Mallory in Figure 6) tries to spoof a particular sender from which she does not have any information. The only available strategy for the attacker is to simply replace the `From` and `Return-Path` headers of the targeted email and try to guess the behavior, composition and transport features.

(*b*) *Known Domain:* In this scenario, the attacker has received or has access to one or more emails sent by a sender that belongs to the same email domain as the spoofed sender. The attacker can thus expect that some of their transport features are present in the emails received by the victim from the sender she wants to spoof.

Table 3: Detection performance of our approach in different threat scenarios.

Threat Scenario	**Blind Spoofing**				**Known Domain**				**Known Sender**			
Algorithm	**kNN**		**SVM**		**kNN**		**SVM**		**kNN**		**SVM**	
Metric	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR
	0.01%	90.9%	0.01%	92.4%	0.01%	72.7%	0.01%	78.1%	0.01%	48.1%	0.01%	30.1%
	0.1%	90.9%	0.1%	92.4%	0.1%	72.7%	0.1%	78.2%	0.1%	48.2%	0.1%	30.2%
	1%	91.1%	1%	92.5%	1%	73.7%	1%	79.3%	1%	48.9%	1%	30.4%
	10%	91.9%	10%	92.9%	10%	78.4%	10%	84.1%	10%	53.2%	10%	33.9%

(*c*) *Known Sender:* In this scenario, shown in Figure 6c, the attacker has received or has access to one or more emails sent by the spoofed sender. As a result, several traits used for constructing the profile are available to the attacker and can be incorporated in her spoofed emails.

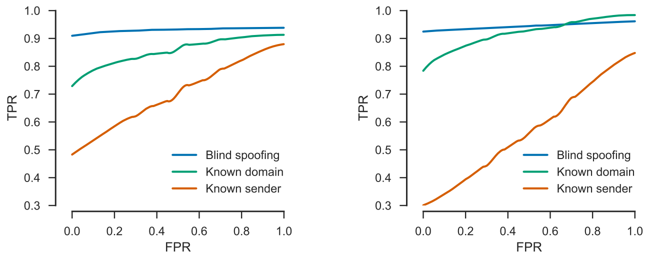
In the following, we describe how we learn a profile of each sender within a mailbox and assign the role of the victim to the owner of the mailbox. Then, based on the attack strategies described in each scenario and using the emails available in our dataset we build corresponding sets of spoofed emails for each sender and combine them with legitimate emails to evaluate our method.

4.3 Spoofed Email Detection

In the following we evaluate the performance of our approach in the threat scenarios defined in the previous section. In order to learn a profile for each sender we begin again by splitting all available emails into training and testing sets. For training, we consider all emails received up to a certain point in time. In the case of the kNN classifier one email from a sender in the training set suffices to make a decision about an incoming email from this origin address, while for the SVM classifier we require a minimum of 5 emails from a sender to include this class during training. In order to tune the parameters of each classifier, we partition the training data into 5 splits and use training/validation partitions, such that the temporal order of emails is preserved—similar to a regular cross-validation. This enables us to simulate training with past data and generating predictions for data yet unseen. Note that although a mailbox or sender may not present enough emails for training, we still use these samples to generate test spoofed emails.

For the testing phase, we combine the test set of legitimate emails with a set of emails crafted according to the attacker strategies described in Section 4.2. In the case of a *blind spoofing* attack, we select a random set of emails sent to recipients at different domains than the victim and label them as the spoofed sender. Likewise, we evaluate the *known domain* attack by selecting emails sent from the domain of the spoofed sender by a different sender to other recipients. Finally, we select emails sent by the spoofed sender to different recipients to built the spoofed test set in the evaluation of the *known sender* attack.

During testing, we expect a legitimate email to be assigned to its true class by the classifier. On the contrary, a spoofed email should be assigned to any of

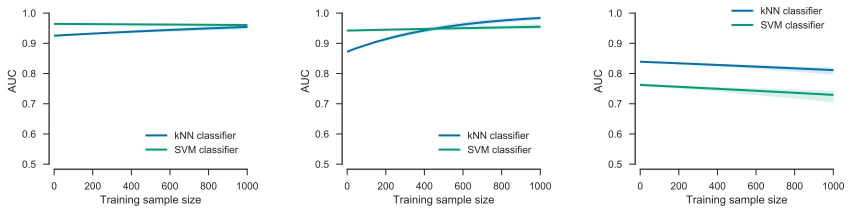


(a) kNN (k=10) **(b)** SVM

Fig. 7: ROC curves for the classification of legitimate emails versus emails spoofed by attackers with different levels of knowledge.

the other classes, resulting in a mismatch between the sender address from which the email is sent and the output of the classifier. There exists thus a trade-off between the probability of detecting a spoofed email and the probability of wrongly highlighting a legitimate email as spoofed. The ROC curves depicted in Figure 7 show the trade-off between the false-positive rate and the false-positive rate for both classifiers.

If the attacker lacks any knowledge about the spoofed sender, we observe that the kNN and SVM classifiers can identify a spoofed email with a true-positive rate of 90.9% and 92.4% respectively at a low false-positive rate of 0.01%. If the attacker has access to emails originating from the same domain, the performance decreases to 72.7% and 78.1% but the classifier is still able to effectively operate at the same low false-positive rate. In the worst-case scenario, the attacker has enough information to craft an email that resembles the learned profile of the spoofed sender, which causes the performance of the classifier to deteriorate considerably. Table 3 specifies numerically the detection achieved at 0.01%, 0.1%, 1% and 10% of false-positive rate for both classifiers in all scenarios.



(a) Blind spoofing **(b)** Known domain **(c)** Known sender

Fig. 8: Area under the ROC curve as a function of the number of training emails used to learn each sender's individual profile.

As mentioned above, we set a lower threshold for the minimum number of emails required to train an SVM classifier. However, as shown in Figure 3 a larger number of emails above this threshold is available for many senders. Figure 8 shows in each scenario the relation between the number of emails from a sender used to train the classifier and the AUC averaged over all mailboxes and senders.

As described in Section 4.1, sender profiles tend to be more compact with an increasing number of emails. However, this can affect the performance differently depending of the knowledge available to the attacker. For instance, in threat scenarios a) and b), emails are classified with an AUC over 0.85 with a small number of training samples. Spoofed emails lay here far enough from the sender profile, leading to a stable or increased performance when classes becomes more populated. In particular, the SVM classifier offers a better performance at a low number of available emails, while with an increasing training size, the kNN classifier surpasses the SVM.

On the contrary, in threat scenario c) the attacker is always able to craft an email that resembles the profile of the spoofed sender, while a larger number of training samples increases the variability of the sender profile. As each spoofed email lay very close or within the target class, it becomes more difficult for the classifier to correctly separate legitimate emails from spoofing attempts when the sample size increases. A possible approach in such a high risk scenario, is to operate the classifier at a higher FPR point and to retrain the model more often on a smaller sample of the most recent emails received from each sender. Finally, the use of a linear SVM for classification allows us to study how the learning algorithm assigns different weights to each type of features according to its importance for the classification. To this end, we determine the distribution of the normalized SVM weights and group them by trait types. In Figure 9, we can observe that, in comparison with behavior and composition features, transport related features manifest both a smaller dispersion and a larger influence on the decision of the classifier. Consequently, transport features have the most discriminative power and, at the same time, are the most difficult traits to forge as even a skilled adversary is not able to fully control transport features without having access to the same delivery infrastructure of the sender.

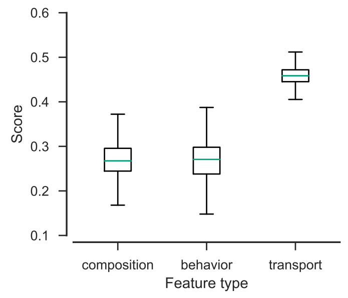


Fig. 9: Distribution of scores per group of traits as learned by the linear SVM classifier during training.

5 Discussion & Limitations

The evaluations in the previous section show that our method is capable of reliably discriminating thousands of senders and identifying spoofed emails if the attacker has limited knowledge of the email structure. Due to the problem setting of detecting spoofing at the receiving side, however, our approach has some inherent limitations which are discussed in the following.

Advanced forgery Although spear-phishing and other targeted email attacks today focus on the forgery of visible features like the sender address, the subject and the content of an email to mimic trustworthy emails [18, 26], we likely have to deal with more advanced attacks in the near future. If current attacks are no longer successful because of increased user awareness and detection approaches like ours, attackers will adapt their techniques. For our method, the best strategy for evasion is to forge as many features from the original sender as possible. An almost perfect forgery is thus a copy of an original mail including also its true transport features as observed by the recipient and enriched with some malicious content. However, the attacker needs to take care of several traits that our method inspects, such as timestamps, IP addresses in received headers and characteristics of the attachment. In the worst case, the attacker is able to forge all of these details and hence the only indication of a spoofed email are minor inconsistencies between IP addresses and hostnames. Our method fails in this scenario, as only a few features differ from the sender model. Nonetheless, the acquisition of emails from a sender and acquiring access to the senders delivery infrastructure to control the transport features, clearly raise the bar for conducting spear-phishing attacks. Therefore and with the current lack of alternative protection approaches, our approach is a valuable extension to current defenses.

Privacy and feature extraction We have implemented the feature extraction in a privacy-friendly way in that all sensitive information of sender, transport and recipients is only stored in an anonymized form by using a hash with random salt. Only these anonymized features are kept and used in the initial creation or retraining of the model. This makes it possible to implement the system for example in a security appliance which receives all feature vectors for analysis but does not store the mails. This also means, however, that the model cannot be simply extended with new features and retrained with old data, since the original mail as input for feature extraction is no longer available. Feature extraction is therefore performed locally in every case. Although this limits how anonymized data from different sources can be combined for analysis, the recipient's email information never leaves the local machine, avoiding privacy issues and possible attack vectors.

Mislabeled data The possibility of the training data containing spoofed emails should not be ignored. However and due to their very nature, the prevalence of spear-phishing emails can only be very low within all emails sent to a recipient. This problem, known as *label noise* [see 8], entails that training samples can be considered subjected to an additive noise during training with a probability of their labels being flipped. In our setup, however, such probability will be very low and the effect during testing of such infrequent examples, while existent, will be negligible.

6 Related Work

The detection of unwanted and malicious emails is a well-established problem in security research. Several methods have been devised in the last years that are related to our approach and which we briefly discuss in the following.

For instance, several approaches exist that focus on the content of emails and the style in which they are written [e.g., 10, 17, 33]. The assumption behind these features is that the writing style of one sender differs significantly from another and that it is too hard for the attacker to write a mail in the same style as the sender she is trying to spoof. The implementation of such content-based features can be as simple as using a 5-gram tokenizer [27] but can also be more complex and include character distributions, atypical words or more advanced stylometric features [10, 17, 33]. In many cases, these stylometric features are used in combination with further behavioral features, such as the time of writing.

While these approaches potentially provide a good detection of spoofed emails, they present two problems. First, if text from the original sender is available from any source stylometric traits can be easy to forge and second such approaches require sufficient data to infer minor differences in stylometry and can be computationally expensive. As a consequence, previous work often operates with small datasets. For example, Lin et al. [27] conduct an evaluation with only 6 senders due to a lack of available data. Similarly, Duman et al. [10] discriminate only 215 senders in their experiments. Whether these techniques can be scaled to cover thousands of senders is unclear and thus the application stylometric features for spear-phishing detection is still an open issue.

The problem of limited learning data is addressed by Stringhini and Thonnard [33] who propose a detection approach that, while also relying on email content, is capable of analyzing larger datasets. However, their method requires a minimum of 1,000 emails per sender to be effective. Moreover, they position the defense at the sender's server and require to include emails from different mailboxes to build a reliable behavioral profile of a user. Such an approach is orthogonal to our method which operates at the recipient's side, who only requires the information contained in her own mailbox to build an effective defense. Furthermore, recipient related features are based on the idea that different recipients have different risk to get spear-phishing mails. Such features are proposed by Amin [1] which determine the amount of information returned by a search engine about a recipient and how often a person has received malicious mails in the past. Unsurprisingly, the latter turns out to be a dominant feature, i.e., those senders who got attacked in the past a lot will probably also get a lot attacked in the future.

As in our work, infrastructure related features commonly include properties of the transport like the senders IP address or her geographic location [17, 27]. But also features of the used mail client belong in this category since a sender will usually use only a single or few email clients. Features related to the infrastructure are often similar for all senders in the same domain which can be used to increase model accuracy when only a few mails from a specific sender are available. Compared to stylometric features, infrastructural features do not model the actual author but only her environment. Therefore, it is impossible to detect a hacked account with these features. On the other hand infrastructural features need less training data to create a well-performing model. Thus, it might be useful to combine the strength of both approaches. Structural based features, instead of content based features are the dominant ones in our evaluation. Such features were already used by [1]. Contrary to this work, our approach makes use of a larger set of features from the mail client and from its transport and is based on distinguishing different senders based on these features instead of globally distinguishing all spear-phishing mails from all benign mails.

Finally, a method recently proposed by Ho et al. [19] focuses on the identification of credential phishing and is designed to identify attacks from unseen senders. Our approach is orthogonal to this work, as it addresses two of its main shortcomings: First, Ho et al. [19] considers the problem of address spoofing irrelevant due to the availability of DKIM and DMARC. Our empirical analysis, however, shows that both techniques are not widely available in practice and thus alternative methods are needed. Furthermore, DKIM and DMARC need to be implemented at the sending side, which enables the attacker to choose a known sender with lacking support for this security feature. Second, the proposed method requires the victim to interact with the phishing email by clicking on a link. This poses a serious security risk and may result in the victim's host being compromised before the attack is actually detected. Our approach does not require interaction and can block phishing attacks before they reach their victim, for example, by removing links and attachments from emails.

7 Conclusions

In this paper, we show that a sender leaves several traits in the structure of an email, resulting from her personal preferences, email client and infrastructure. Based on these traits, we present a detection method that is capable of learning profiles for senders and identifying impersonated emails without relying on its content or server-side implementations. In an empirical evaluation with over 17,000 senders, we demonstrate that this method can identify over 90% of spoofed emails with less than 1 false alarm in 10,000 emails, if the attacker has no knowledge of the sender's profile. If the attacker has access to emails from the same domain as the spoofed sender our method still attains a detection rate of 72% and thus raises the bar for an adversary to effectively complete a spoofing attack. Although our approach cannot detect an attack by an adversary with vast resources, it provides a strong protection from attackers that are not able to obtain original emails from a specific sender. In practice, our approach thus provides a valuable tool for fending off spear-phishing attacks that would go unnoticed without a proper anti-spoofing detection.

Bibliography

[1] R. M. Amin. *Detecting Targeted Malicious Email Through Supervised Classification of Persistent Threat and Recipient Oriented Features.* PhD thesis, George Washington University, Washington, DC, USA, 2010. AAI3428188.

[2] A. Beygelzimer, K. S., and J. Langford. Cover trees for nearest neighbor. In *International Conference on Machine Learning (ICML)*, pages 97–104, 2006.

[3] BuildWith Website. Buildwith technology lookup. https://builtwith.com, visited November, 2017.

[4] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. OpenPGP Message Format. RFC 4880 (Proposed Standard), Nov. 2007. ISSN 2070-1721. Updated by RFC 5581.

[5] D. D. Caputo, S. L. Pfleeger, J. D. Freeman, and M. E. Johnson. Going spear phishing: Exploring embedded training and awareness. *IEEE Security & Privacy*, 12(1):28–38, 2014.

[6] P. Chen, L. Desmet, and C. Huygens. A study on advanced persistent threats. In *IFIP International Conference on Communications and Multimedia Security*, pages 63–72. Springer, 2014.

[7] D. Crocker, T. Hansen, and M. Kucherawy. DomainKeys Identified Mail (DKIM) Signatures. RFC 6376 (Internet Standard), Sept. 2011. ISSN 2070-1721.

[8] N. D. Lawrence and B. Schölkopf. Estimating a Kernel Fisher Discriminant in the Presence of Label Noise. In *ICML*, volume 1, pages 306–313, 2001.

[9] R. Duda, P.E.Hart, and D.G.Stork. *Pattern classification.* John Wiley & Sons, second edition, 2001.

[10] S. Duman, K. K. Cakmakci, M. Egele, W. Robertson, and E. Kirda. EmailProfiler : Spearphishing Filtering with Header and Stylometric Features of Emails. In *COMPSAC*, 2016.

[11] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[12] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[13] I. D. Foster, J. Larson, M. Masich, A. C. Snoeren, S. Savage, and K. Levchenko. Security by any other name: On the effectiveness of provider based email security. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 450–464, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3832-5. https://doi.org/10.1145/2810103.2813607.

[14] N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. RFC 2045 (Draft Standard), Nov. 1996. ISSN 2070-1721. Updated by RFCs 2184, 2231, 5335, 6532.

[15] N. Freed and K. Moore. MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations. RFC 2231 (Proposed Standard), Nov. 1997. ISSN 2070-1721.

[16] S. Gupta, A. Singhal, and A. Kapoor. A literature survey on social engineering attacks: Phishing attack. In *Computing, Communication and Automation (ICCCA), 2016 International Conference on*, pages 537–540. IEEE, 2016.

[17] F. Han and Y. Shen. Accurate Spear Phishing Campaign Attribution and Early Detection. In *SAC*, pages 2079–2086, 2016.

[18] S. Hardy, M. Crete-Nishihata, K. Kleemola, A. Senft, B. Sonne, G. Wiseman, P. Gill, and R. J. Deibert. Targeted threat index: Characterizing and quantifying politically-motivated targeted malware. In *USENIX Security*, pages 527–541, 2014.

[19] G. Ho, U. C. Berkeley, A. Sharma, T. Lawrence, B. National, M. Javed, U. C. Berkeley, V. Paxson, U. C. Berkeley, D. Wagner, U. C. Berkeley, G. Ho, and A. Sharma. Detecting Credential Spearphishing Attacks in Enterprise Settings. In *USENIX Security Symposium*, 2017. ISBN 9781931971409.

[20] T. M. Inc. Spear-Phishing Email: Most Favored APT Attack Bait. Technical report, Trend Micro Inc., 2012.

[21] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. Technical Report 23, LS VIII, University of Dortmund, 1997.

[22] T. Joachims. *Learning to classify text using support vector machines: Methods, theory and algorithms.* Kluwer Academic Publishers, 2002.

[23] S. Josefsson. The Base16, Base32, and Base64 Data Encodings. RFC 4648 (Proposed Standard), Oct. 2006. ISSN 2070-1721.

[24] S. Kitterman. Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1. RFC 7208 (Proposed Standard), Apr. 2014. ISSN 2070-1721. Updated by RFC 7372.

[25] M. Kucherawy and E. Zwicky. Domain-based Message Authentication, Reporting, and Conformance (DMARC). RFC 7489 (Informational), Mar. 2015. ISSN 2070-1721.

[26] S. Le Blond, A. Uritesc, and C. Gilbert. A look at targeted attacks through the lense of an ngo. In *USENIX Security*, pages 543–558, 2014.

[27] E. Lin, J. Aycock, and M. Mannan. *Lightweight Client-Side Methods for Detecting Email Forgery*, pages 254–269. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-35416-8. https://doi.org/10.1007/978-3-642-35416-8_18.

[28] T. Mori, K. Sato, Y. Takahashi, and K. Ishibashi. How is e-mail sender authentication used and misused? In *Proceedings of the 8th Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference*, CEAS '11, pages 31–37, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0788-8. https://doi.org/10.1145/2030376.2030380.

[29] B. Ramsdell and S. Turner. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification. RFC 5751 (Proposed Standard), Jan. 2010. ISSN 2070-1721.

[30] P. Resnick. Internet Message Format. RFC 5322 (Draft Standard), Oct. 2008. ISSN 2070-1721. Updated by RFC 6854.

[31] K. Rieck, C. Wressnegger, and A. Bikadorov. Sally: A tool for embedding strings in vector spaces. *Journal of Machine Learning Research (JMLR)*, 13(Nov):3247–3251, Nov. 2012.

[32] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[33] G. Stringhini and O. Thonnard. That Ain't You: Blocking Spearphishing Through Behavioral Modelling. In *DIMVA*, 2015.

[34] J. Wang, T. Herath, R. Chen, A. Vishwanath, and H. R. Rao. Research article phishing susceptibility: An investigation into the processing of a targeted spear phishing email. *IEEE Transactions on Professional Communication*, 55(4):345–362, 2012.

A Appendix

Tables 4, 5 and 6 provide an overview of the different traits characterizing the behavior, composition and transport of emails, respectively.

Table 4: List of behavior features.

Identifier	Cardinality	Description	Examples
attachment-type	n	Type of attachment	attachment-type(image)
hdr-empty	n	Headers with empty values	hdr-empty(cc)
hdr-local-domain	n	Headers indicating local domains	hdr-local-domain(to:D0)
hdr-related-mails	n	Headers indicating relation to other emails	hdr-related-mails(subject:re)
hdr-count	n	Number of standard headers and their values	hdrcount(cc:1:2+)
hdr-x	n	Occurrences of non-standard headers	hdr-x(x-virus-scanned)
msgid	n	Structural description of Message-Id header	msgid(<A.A@H>)
reply-to	n	Hashed sender in Reply-To header	reply-to(xxx)
resent	1	Headers indicate redistribution	resent(1)
return-path	n	Sender in Return-Path header	return-path(full:same-as-from)
text-quoted	1	Ratio of quoted total text in main part	text-quoted(0.3)
frompart	n	2-grams of From field	frompart(xxx:yyy)
from	n	Multiple senders in From header	from(full:*)

Table 5: List of composition features.

Identifier	Cardinality	Description	Examples
base64	n	Peculiarities of Base64 transfer encoding	base64(linelen(72))
quoted-printable	n	Peculiarities of Quoted-Printable transfer encoding	quoted-printable(unencoded-ctrl)
7bit	n	Peculiarities of 7bit transfer encoding	7bit(7bit-contains-8bit)
8bit	n	Peculiarities of 8bit transfer encoding	8bit(long-line)
attachment-ext	n	Extension of the attachment	attachment-ext(doc)
attachment-mism	n	Mismatch of attachment type and extension	attachment-mism(doc|zip)
attachment-sig	1	Signature of how the attachment is specified	attachment-sig(fTnT)
inline-ext	n	Extension of attachment when disposition inline	inline-ext(jpeg)
nodisposition-ext	n	Extension of attachment if no disposition is given	nodisposition-ext(jpeg)
boundary	n	Structural description of the MIME boundary	boundary(-=_H-H-H)
hdr-syntax	n	Syntactic format of headers	hdr-syntax(subject:q:ISO-8859-1)
hdr-pair	n	Pair-wise order of headers	hdr-pair(from:date)
part-hdr-pair	n	Pair-wise order of headers in MIME parts	part-hdr-pair(content-type:content-id)
ua	n	Simplified name of user agent	ua(outlook16)
preamble	n	Digest of the MIME preamble	preamble(c928c8bf)
mime	n	Peculiarities of MIME usage	mime(cd:inline+filename)
depth	1	Depth of the MIME structure	depth(2)
mime-warning	n	Minor problems in MIME structure	mime-warning(invalid-content-type)
mime-error	n	Major problems in MIME structure	mime-error(paramval-junk)
part-path	n	Path to MIME parts	part-path(alt(R).1:html)
part-size	n	Size of MIME parts	part-size(html:10:1000)
part-type	n	Type of MIME parts	part-type(image:base64)

Table 6: List of transport features.

Identifier	Cardinality	Description	Examples
dkim	n	Results of DKIM validation	dkim(1:valid), dkim(2:invalid)
rcvd	1	Number of Received headers	rccvd(13)
rcvd-pair	n	Hashes of previous and current Received header	rcvd-pair(xxx:yyy)
rcvd-mta	n	Hashes of MTA features at given header position	rcvd-mta(1:XXX)
rcvd-src	n	Hashes of source features at given header position	rcvd-src(2:xxx)
rcvd-tls	n	Hashes of TLS features at given header position	rcvd-tls(3:xxx)
rcvd-tocc	n	Occurrences of To field in Received headers	rcvd-tocc(to:x1)
hdrtz	1	Path of time zones from Received headers	hdrtz(-0200:+0800)
hdrtzcost	1	Cost of transport based on the changes in time zones	hdrtzcost(6)
srcip-asn	1	ASN for source IP address of client	srcip-asn(8881)
srcip-spf	1	SPF result for source IP address of client	srcip-spf(spf:Pass)