Birds of a Feather Flock Together: The Accidental Communities of Spammers

Yehonatan Cohen
Computer Science Department
Ben Gurion University of the Negev
Be'er Sheva, Israel
Email: yehonatc@cs.bgu.ac.il

Danny Hendler
Computer Science Department
Ben Gurion University of the Negev
Be'er Sheva, Israel
Email: hendlerd@cs.bgu.ac.il

***Abstract*—Outbound spam email is a serious issue for Email Service Providers (ESPs). If not resolved, or at least sufficiently mitigated, ESPs may incur higher costs and suffer damage to their reputation. In this work, we investigate the early detection of spamming accounts hosted by ESPs. Our study is based on a large real-life data set, consisting of mail logs involving tens of millions of email accounts hosted by a large, well-known, ESP.**

An analysis of our data set reveals that spammers tend to be clustered in the same communities within the graph induced by inter-account email communication. The reason for this phenomenon is, most likely, that spammers often use the same techniques for harvesting email addresses. As a result, they inadvertently spam each other or spam the same legitimate accounts. We leverage this accidental community structure for devising a highly accurate spammer detector that outperforms previous algorithms by a wide margin.

I. Introduction

With almost 4 billion accounts worldwide in 2013 and an expected average annual growth of 6%, email is probably the most popular Internet-based application [28]. Unfortunately, email has become a fertile ground for the distribution of *spam mail*, consisting of unsolicited messages sent in bulk, mostly of advertisement contents, but often used also for spreading malware or conducting phishing attacks. According to recent statistics, approximately 70% of worldwide email traffic during 2013 was spam [17]. *Outbound spam* poses a serious problem to Email Service Providers (ESPs).

Often originating from zombie computers and compromised accounts within their infrastructure, outbound spam may hurt ESPs in several ways. First, vast amounts of email being sent from ESP domains overload ESP servers and communication infrastructure [11]. In addition, ESPs from which large numbers of spam messages are sent are likely to become blacklisted, thereby preventing the legitimate users of these ESPs from exchanging email and disconnecting them from external domains. Indeed, ESPs that fail to deploy effective outbound spam filtering mechanisms provide poor user experience and thus hurt their popularity and reputation [32]. A number of techniques for detecting outgoing spam have been developed and are widely-used. *Content-based filters*, i.e. filters that learn and identify textual patterns of spam messages, are used by most ESPs [20]. Unfortunately, spammers have developed their own techniques for evading content-based filters, such as using image-spam mails [1] and hash-busters [21]. Thus, complementary detection techniques are required for identifying spammer accounts some or all of whose messages go undetected by filters. The analysis of inter-account communication is such a technique.

Whereas content-based filters consider the properties of individual messages, the analysis of the social network induced by inter-account communication can reveal the patterns of social interactions between email accounts. These patterns are typically very different for spammer and non-spammer accounts [6], [13], [16], [22], [24], [33]. Inter-account communication is modeled using a graph in which a node is generated for each email account appearing in the data set and an edge connecting two nodes appears if and only if there was email exchange between the two corresponding accounts [16].

Our Contributions: Our study uses a large real-world data set, consisting of outgoing mail logs involving tens of millions of email accounts hosted by a large, well-known, ESP. By analyzing this data set, we observe that spammers tend to be clustered in the same communities within the graph induced by inter-account email communication. As we explain in Section IV, this is an unintentional artifact of the manner in which spammers harvest email addresses. We identify different types of inter-account communication patterns that cause the formation of these accidental spammer communities.

Our goal is to achieve *early detection*, that is, to devise an algorithm that detects spammers before their messages are detected by a content-based filter and possibly even if they are not detected by the filter at all. We leverage the existence of spammer communities for devising a highly efficient early detection algorithm. Our empirical evaluation establishes that our detector outperforms previous algorithms by a wide margin.

II. Background and Related Work

A. Community Detection

It is well-known that networks that represent social interactions tend to exhibit *community structure* [15]. In a nutshell, community structure means that the network may be clustered to sets of nodes, each of which relatively densely-interconnected internally, with (relatively) few connections between different sets. In the core of most community detection algorithms lies a *quality function* that assigns a score to a given clustering of the graph. *Modularity* [25] and *Conductance* [7] are two widely-used quality functions.

The Louvain algorithm [12] is a widely-used community-detection algorithm for large-scale networks. It is an iterative algorithm that greedily attempts to maximize the modularity of a graph partition. It starts with an initial partitioning, where each partition contains a single node, and in each iteration attempts to maximize modularity by changing the assignments of sub-communities into higher level communities. The algorithm halts when it reaches a local maximum. The default output of the Louvain implementation that we use [18] is the partitioning corresponding to the local maximum. However, similarly to other hierarchical clustering algorithms, it also returns a *dendrogram* - which is a hierarchical structure representing all graph partitions reached throughout the execution of the community detection algorithm.

B. Email Spam/Spammer Detection

Two key approaches are used to detect spam mails and spamming accounts in email networks: content-based and topology-based. Essentially, content-based approaches rely on the fact that spam messages differ from legitimate messages in the vocabulary used by them and in the URLs they embed.

Topology-based approaches, on the other hand, rely on structural features derived from inter-account communication patterns. Early topology-based works made use of several known features and utilized them to separate spamming accounts from legitimate ones. Boykin and Roychowdhury [6] leveraged the fact that spammers usually spam arbitrarily-chosen email addresses which are not likely to know each other. Their email spam detection scheme relies on the Clustering Coefficient (CC) feature.

Lam and Yeung [22] used a set of simple features, including the in/out-degree of a user and its numbers of incoming/outgoing mails, along with more sophisticated features like CC and communication reciprocity (CR) - the fraction of accounts with which an account had bi-directional communication out of the accounts to which it sent emails.

An identical set of features was used by Tseng and Chen [33], whom proposed an incrementally-learning spam detector. In a later study, Cohen et al. [13] defined sender account features and presented a detector that combines a content-based filter (used for model training) and topological features.

Moradi et al. [8] evaluated the quality of several common community detection algorithms on large-scal email traffic by examining the level of homogeneity in the resulting structure, that is, the percentage of communities containing legitimate users only, communities consisting of spamming accounts only, and mixed communities.

C. Clustering-Based Spammer Detection

Several prior spammer detection studies employ community detection. Li and Hsieh [23] propose to cluster email senders based on common URLs appearing in their emails. They generate a graph where nodes represent senders and an edge is introduced between a pair of senders if there is a URL that appears in an email sent by both. They observe that spammers are highly clustered in the resulting graph. Xu et al. [9] cluster spammers based on their behavior during the harvesting phase, in which the email addresses which they spam are collected.

Our approach and those of [23], [9] differ in terms of the graphs they construct and the data they require. Whereas our approach is based on the graph of direct inter-account communication, [23], [9] are based on similarity graphs that require data that is often hard to obtain: [23] requires access to message contents and [9] requires data pertaining to the email address harvesting phase.

Bhat and Abulaish [3] propose to use overlapping community structure (computed by OCTracker [2]) for obtaining community-based features, such as the number of communities to which a node was assigned and the total number of foreign nodes to which a node has out-links. A machine-learning model is trained based on these features. Their evaluation is based on Facebook wall-posts [34] and on Enron's data set [30], combined with a set of synthetic spammers.

In another study, Fire et al. [14] partitioned social-networks using the Louvain algorithm. They defined several features based on this partitioning, such as the number of communities to which an account belongs and the ratio between the number of connections between an account's friends and the average number of friends within communities. Their trained machine-learning model can be then used to detect fake profiles and spammers.

Whereas [3], [14] employ social graph clustering for deriving useful features regarding the communities to which an account is connected, our detector directly leverages the fact that spammers tend to cluster in the same communities for improved detection accuracy. We compare the performance of our detector with that of [3], [14] (as well as that of additional detectors) on a large real-life data set.

The rest of this paper is organized as follows. We describe our data set in Section III. We analyze the dynamics of spamming communities in Section IV. This is followed by a description of our algorithm in Section V. We report on the results of our empirical evaluation in Section VI. Section VII concludes the paper with a short discussion of our results.

III. Data Set

Our data set mostly consists of outgoing mail logs, storing log records of outbound email messages. Outbound messages are, in general, relayed to their destinations and generate log lines that are written to the outgoing mail logs. Messages whose destinations are internal accounts (i.e. accounts hosted by the ESP) are relayed by an outgoing mail forwarding server, whereas messages sent to external accounts are relayed by an outgoing mail server. Outbound messages are processed by a content-based spam detector that tags them as either spam or ham (non-spam). Whether or not messages tagged as spam are filtered out depends on the receiver's identity: the ESP has different contracts with different customers, which in some cases mandate that spam messages should still be relayed to the customer, with a "Spam" tag appended to the message subject; in other cases, spam messages are simply discarded but are nevertheless logged.

The data set consists of log records collected over a time period of 26 days, during August 2010. Table I compares the data set used by this work with the largest data sets used in previous studies of outbound spam detection.[1] Whereas previous studies of outgoing spam mostly use small-sized datasets taken from academic institutes [16], [33] or Enron's public data set [22], [3], our data set was collected by the mail servers of a large ESP.

We designate an account as a *spamming account* if, according to the outgoing logs, it sent one or more messages tagged as spam by the content-based filter. Clearly, spamming accounts are internal. The data set contains $4.6E6$ internal accounts, of which $14,209$ are spammers, and $3E7$ external accounts. $1.3E8$ email messages are recorded by the logs, of which $2.8E6$ were tagged as spam by the content-based detector. The log files in the data set contain meta-data regarding exchanged emails, such as the date and time of delivery, the IPs they were sent from[2], etc., but *do not contain any information regarding the contents of the email*, except for a spam/ham tag. Each internal account has a unique identifier associated with it, which appears in every log record that corresponds to a message sent by it, regardless of the address used by the account in the "From" field when sending the email. The data set also contains incoming log files, storing records of inbound email messages, logged over a period of 4 days. These, however, are not used by our algorithm.

IV. Spammer Communities

An analysis of the data set reveals that spammers often spam each other. This motivated us to check whether spammers hosted by the ESP are clustered in the same communities.

We partition $G_d = <V_d, E_d>$, the daily communication graph of day d, using the Louvain algorithm for community detection [12].[3] The Louvain algorithm can partition large social networks with millions of users in a reasonable period of time. Let $spam(a, d)$ denote the number of spam mails sent by account a during day d. We compute the *spamminess* of each community C discovered in G_d (denoted by $\mathcal{S}_d(C)$) according to Equation 1, as the fraction of internal accounts in C that have sent one or more spam messages during day d. We note that only internal accounts can be labeled as spammers, since G_d is generated based on outgoing log files. However, the graph also includes nodes representing external addresses to which messages were sent.

$$\mathcal{S}_d(C) = \frac{\big|\{a \in C | spam(a, d) > 0\}\big|}{\big|C\big|}. \tag{1}$$

Figure 1 presents both the distribution-histogram and the complementary cumulative distribution function (ccdf) graph of spammers over the spamminess of their communities. Although the overall rate of spammers in the data set is only 0.1% (see Section III), 83% of spammers are found in communities with spamminess higher than 0.15 and almost 30% in communities with spamminess 0.9 or higher.

What are the reasons that spammers tend to cluster in the same communities? An in-depth analysis of spamming communities found in the data set reveals that there are three key communication patterns that drive this phenomenon. Figures 2 - 4 (prepared using Pajek[4]) show 3 communities found in the data set that illustrate these patterns.

Spammers that spam each other: An analysis of the data set reveals that a large portion of a spammer's incoming mail is spam, in sharp contrast to the incoming mail of legitimate accounts. On average, 32% of emails received by spammers were sent by other spammers, whereas the corresponding figure for legitimate accounts is only 0.3%. We explain this observation as follows. Legitimate email users seldom send mails to spammers, whereas email address harvesting techniques such as dictionary attack [5], on the other hand, may cause spammers to spam each other. Figure 2 depicts a sample community $C \in G_d$ in the data set, where multiple spammers spam other spammers. Nodes labelled as 'Spammer' represent accounts that have sent spam messages on day d. Nodes labelled as 'Early Detected' represent nodes that have only sent spam in later days but were identified by our detector on day d (see Section V). None of the messages sent by nodes labelled as 'Normal' were tagged as spam by the content-based filter. Clearly, external accounts are always labelled as 'Normal'.

Common victims: In some communities, we observe a communication pattern in which legitimate email accounts receive spam mail from several spammers (see Figure 3). We explain this observation as follows. First, bots from the same botnet may target the same victim in different times. Additionally, spammers may target the same account even when they belong to different botnets. Indeed, John et al. [20]

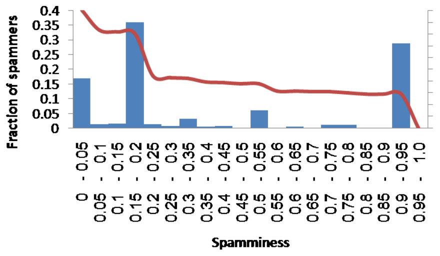

Fig. 1. Spammers distribution over their communities' spamminess values (histogram w.r.t left-hand side y-axis) and corresponding CCDF (red graph, w.r.t. right-hand side y-axis).

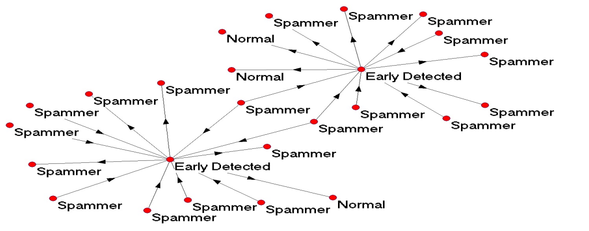

Fig. 2. A community in which spammers spam each other.

[1]The details of additional, smaller, such data sets are presented in [24].
[2]The source IP field is anonymized
[3]We used the implementation of Python's Networkx library [18].

TABLE I. This data set vs. data sets used by previous studies.

	This data set	SUNET[24]	NTU[33]	Enron[3], [22]
#mails	138M	24M	2.86M	0.5M
#edges	55M	21.6M	-	0.36M
#accounts	34M	10.5M	0.63M	0.036M
time period	26 days	14 days	10 days	3.5 years
contents	spam & ham	spam & ham	spam & ham	ham

measured the mailing lists overlap between different spamming botnets and found that some of them share more than 20% of their recipients. These may be botnets that use the same email address harvesting techniques.

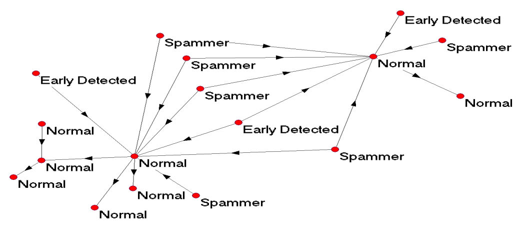

Fig. 3. A community in which spammers share their victims.

Nearby victims: A third communication pattern is revealed by communities where spamming accounts are clustered together because they send messages to inter-connected legitimate accounts. Figure 4 depicts such a community, where spammers on the left-hand side are clustered with spammers on the right-hand side via communication chains consisting of 3 normal accounts. It is well-known that spamming botnets crawl the web in order to harvest mailing lists and email addresses [31], [19], [10]. We hypothesize that this and the use of dictionary attacks [5] are the dynamics behind the formation process of this type of spamming communities. For instance, consider a harvester that crawls the web and harvests email addresses of some organization. The resulting email address list is then partitioned and sub-lists are assigned to different bots [27]. Since organization members are likely to communicate with each other either directly or indirectly, communities such as the one depicted by Figure 4 result.

We note that some communities may exhibit combinations of the above inter-account communication patterns. Moreover, accounts none of whose messages were detected as spam by the content-based filter are not necessarily legitimate; most probably, some of these accounts simply manage to evade the content-based filter all throughout the period of time logged by the data set. Consequently, the distinction between different types of these patterns is fuzzy. For instance, what may seem as a communication pattern of the second type, may actually be a communication type of the first type if some accounts entirely evade the filter. Luckily, by employing community detection, our detector manages to identify spammers that are clustered by all of the above (and possibly additional) inter-account communication patterns.

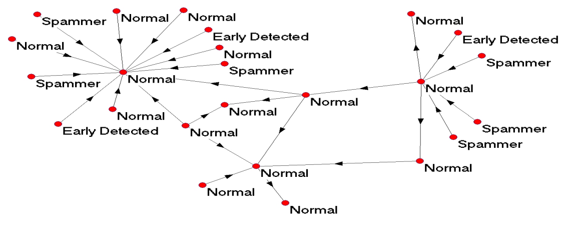

Fig. 4. A community in which spammers targeted nearby legitimate nodes.

V. Detection Scheme

Our approach for the early detection of outgoing spammers leverages the tendency of spammers to cluster within the same communities. The communication patterns involving spammers described in Section IV indicate that allegedly-legitimate accounts within high-spamminess communities are likely to be spammers that evaded detection by the content-based filter.

Indeed, the correlation in the data set between the spamminess of a day-d (for some d) community C and the fraction of C's accounts that do not spam on day d or before but spam on a later day is more than 0.7.

For instance, consider two communities, C_1 and C_2, with spamminess levels of 0.8 and 0.05 respectively. As we show in Section VI, an allegedly-legitimate account of C_1 is much more likely to send spam in the future than an allegedly-legitimate account of C_2. Thus, finding high-spamminess communities is key for the early detection of spammers.

Figure 5 presents the flow of our outgoing spammer detector. The detector receives as its input the outgoing mail logs of a single day. It outputs a list of suspect accounts of size determined by a configuration parameter (denoted K).

We now describe the algorithm in detail (pseudo-code is presented in Algorithm 1). The algorithm is composed of 3 phases. In the first phase, a representation of a daily communication graph, G_d, is constructed based on input mail logs. Each node of G_d represents an email account. Directed edge $(u, v) \in G_d$ signifies that one or more email messages were sent from email account u to email account v during day d. Accounts one or more of whose messages were identified as spam by the content-based filter are tagged as spammers. The rest of the accounts are considered allegedly-legitimate.

The second phase of the algorithm is called the *Spammer-Communities Detector* (SCD). The SCD partitions the communication graph to communities and sorts these communities in decreasing order of their spamminess. A third algorithm phase, called the *Account-Level Detector* (ALD), sorts the accounts of the most suspicious communities in decreasing order of suspicion by using an ensemble machine learning classifier [29] that uses per-account features.

Next, we provide more details about the SCD and ALD.

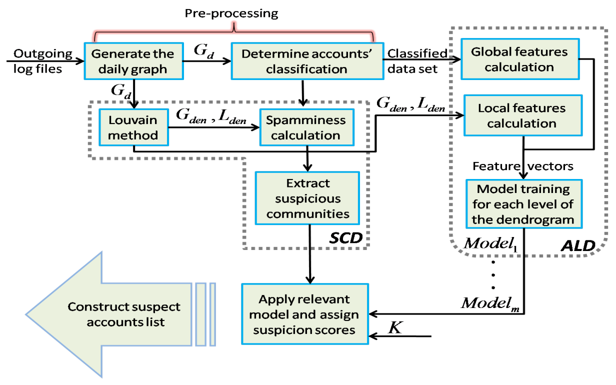
Pre-processing
Outgoing log files
Generate the daily graph
G_d
Determine accounts' classification
Classified data set
Global features calculation
G_d
Louvain method
G_{den}, L_{den}
Spamminess calculation
G_{den}, L_{den}
Local features calculation
Feature vectors
Extract suspicious communities
Model training for each level of the dendrogram
SCD
$Model_1$
ALD
$Model_m$
Construct suspect accounts list
Apply relevant model and assign suspicion scores
K

Fig. 5. Flow diagram of our detector.

A. Spammer-Communities Detector

The SCD implements the second phase of our detector. It uses the Louvain algorithm. The output of hierarchical clustering algorithms in general, and the Louvain algorithm in particular, is represented by a *dendrogram*. Each dendrogram cluster represents a community, generally composed of other sub-communities. The communities of the ith level are the building blocks for communities of level $i+1$.

We leverage the dendrogram's structure and make use of the communities which are encapsulated in it. Let G_{den} denote the dendrogram. The SCD traverses G_{den} and generates a list, L_{den}, containing the communities of all levels. The spamminess of all communities is then computed (see Equation 1). In the dendrogram shown in Figure 6, $\mathcal{S}_d(A) = 0.5$, $\mathcal{S}_d(D) = 0$, $\mathcal{S}_d(E) = 2/3$, etc.

Next, L_{den}'s communities are inserted into a max heap in which items are sorted based on their spamminess values. It is important to note that each node may be present in several communities of L_{den}, effecting their spamminess values.

The SCD outputs the shortest prefix of the list of most suspicious communities in L_{den} whose union contains at least $\alpha \cdot K$ distinct allegedly-legitimate accounts, where $\alpha > 1$ is an implementation parameter (in our current implementation, $\alpha = 10$). This ensures that the input to the account-level detector is a list that is significantly longer than K.

As an example, consider once again the dendrogram of Figure 6. For $K = 2$ and $\alpha = 1.5$, the SCD would return communities E and A. Note that community C contributes no suspects since it contains spamming accounts only, and is therefor ignored by the algorithm.

B. Account-Level Detector

The goal of the third phase is to sort the accounts of the most suspicious communities according to their individual behavior, in decreasing suspicion order, and to extract the K most suspicious accounts.

For each community and for each of its accounts, we compute two sets of simple, lightweight features. The first set consists of the total numbers of an account's distinct senders and recipients and its total numbers of sent/received mails during day d. The second set is the *local* equivalent: The same features are calculated with respect to the node's community, i.e. we ignore messages destined to/originated at nodes from other communities. As mentioned earlier, a node may be associated with several communities from several levels of the dendrogram. In this case, there is a feature vector associated with the account for each of these communities.

Then, for each level i of the daily dendrogram, we train a machine learning model whose goal is to separate spammers and legitimate accounts *in level-i communities*. Model training employs *undersampling* and is done as follows. We pick all of the spamming nodes in our daily outgoing log file and an equal number of randomly chosen legitimate (internal) accounts.[4] These accounts, along with their feature vectors with respect to the level-i communities to which they belong, are used as the training set for a machine-learning algorithm. Undersampling of the legitimate accounts population is required as the data set exhibits imbalance, with 3 orders of magnitude more legitimate accounts than spammers. If not dealt with by undersampling, this imbalance is likely to cause a bias towards legitimate accounts. We use the Rotation Forest ensemble classifier [29], implemented in Weka [26], as our machine-learning algorithm.

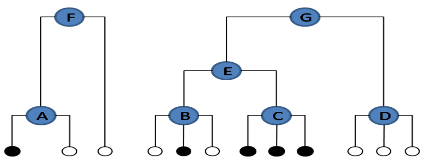

Fig. 6. A sample dendrogram. Spamming nodes are colored black.

The resulting Rotation Forest models are then applied to the accounts of the top suspect communities in the respective levels. Each of these accounts is being assigned a suspicion score, based on which the accounts that appear in the list of communities produced by the SCD are sorted in decreasing suspicion order. Finally, the K most suspicious accounts are returned as the detector output.

VI. Evaluation

In this section, we describe the experiments we conducted to validate the effectiveness of our detector. We compare our detector with the following 5 previously-published outgoing spam detection schemes. Bhat and Abulaish's algorithm [3] (see Subsect. II-C) is based on a machine learning model, trained on a set of topology-related features, some of which are community-based.[5] We refer to their algorithm as *BA-J48*. The algorithm proposed by Fire et al. [14] (see Subsect. II-C) is referred to below as *Fire-J48*.

Lam and Yeung [22] presented a spammer detector that uses a number of features based on inter-account social interactions, such as CR and CC, and utilizes the k-nearest-neighbors algorithm for classification. We refer to their algorithm as *LY-kNN*. Tseng and Chen [33] proposed *MailNET*, an algorithm that uses features similar to those of the *LY-knn* algorithm, but learns only from *pure accounts*, i.e. accounts which send only spam or only ham emails. Their algorithm uses the SVM classifier. Cohen et al. [13] presented *ErDOS*, an algorithm for the early detection of outgoing spammers. ErDOS uses light-weight features, such as the ratio between an account's numbers of incoming and outgoing messages, and employs the Rotation Forest classifier.

A. Evaluation Metrics

Let a be an account in a suspect accounts list produced by a detector on day d. Account a is an *early detected* account, if no message sent by a before or during day d is tagged by the content-based filter as spam, but at least one message sent by a on a later day is tagged as spam.

Account a is a *detected* account, if no message sent by a during day d is tagged by the content-based filter as spam, but at least one message sent by a on another day (either before or after d) is tagged as spam.

[4]Note that the accounts active during day d appear in each level of day d's dendrogram.

[5]Our implementation of their work is based on the J48 decision tree, as it performed better than other machine-learning algorithms.

```
Input  : Daily log files of email transactions
Output: Suspicious accounts list
// Pre-processing:
Generate the daily communication graph,
$G_d = (V_d, E_d)$
foreach email account $v \in V_d$ do
    set $v$'s ham/spam label according to content-based filter
end
// Spammer-Communities Detector:
$G_{den} \leftarrow Louvain(G_d)$
$L_{den} \leftarrow$ List of all communities in $G_{den}$
$Communities \leftarrow$ empty max heap, sorted by $spamminess$ values
$TopSuspectCommunities \leftarrow \emptyset$
$SuspectsSet \leftarrow \emptyset$
foreach community $C \in L_{den}$ do
    $spamminessVal \leftarrow C$'s spamminess value
    Communities.Insert($spamminessVal$,$C$)
end
while $|SuspectsSet| < \alpha \cdot K$ do
    C $\leftarrow$ Communities.Extract-Max()
    $TopSuspectCommunities \leftarrow TopSuspectCommunities \cup \{C\}$
    $SuspectsSet \leftarrow SuspectsSet \cup \{v | v \in C$ & $v$ is an allegedly legitimate account $\}$
end
// Account-Level Detector:
$SuspectsList \leftarrow$ an empty list
foreach email account $v \in V_d$ do
    $G(v)$ = Global-Features($v$)
end
foreach community $C \in L_{den}$ do
    foreach email account $v \in C$ do
        $F(v)$ = Local-Features($v$,$C$)
    end
end
foreach level $i$ of $G_{den}$ do
    $VectorsSet_i = \{(G(v), F_C(v)) \mid level(C) = i,\ v \in C\}$
    training set = Undersampling($VectorsSet_i$)
    $Model_i \leftarrow$ ROTATION-FOREST(training set)
end
Accounts $\leftarrow$ max heap sorted by values assigned by classification model, initially empty
foreach community $C \in TopSuspectCommunities$ do
    $Model = Model_{level(C)}$
    foreach account $a \in C$ do
        score $\leftarrow Model(a)$
        Accounts.Insert(score,$a$)
    end
end
while $|SuspectsList| < K$ do
    $a \leftarrow$ Accounts.Extract-Max()
    if $a \notin SuspectsList$ then
        SuspectsList.append($a$)
    end
end
return SuspectsList
```

Algorithm 1: Early detection of outgoing spammers.

We use the following evaluation metrics:

Early Precision (e-Precision): This is the percentage of accounts in the detector's daily suspect accounts list that are early-detected accounts.

Precision: This is the percentage of accounts in the detector's daily suspect accounts list that are detected accounts.

Enrichment factor (EF): When applied to e-Precision, EF compares the e-Precision of a list of suspicious accounts returned by a detector with that of a randomly generated list. More formally: EF for day d compares the e-Precision of the list produced by a detector for day d with that of a list of the same length whose accounts are randomly selected from the entire population of email accounts that have sent no messages tagged as spam up to (and including) day d.

$$EF = \frac{\text{e-Precision}(detector\ list)}{\text{e-Precision}(random\ list)}. \tag{2}$$

The higher the score, the stronger is the indication of a large proportion of early-detected accounts in the detector's list in comparison with a random list. The EF metric can be defined similarly for evaluating the (non-early) precision quality of a detector list.

B. Experiments and Results

Figure 7 presents the e-Precision of each of the algorithms as a function of the suspects list length, averaged over the first 10 days of the 26 days period logged by the data set. It can be seen that regardless of the list length, the new algorithm significantly outperforms the other algorithms. For list length of 100, the new algorithm achieves e-Precision of almost 35%, approximately 7 times higher than the rest of the algorithms. As the list length is increased, the e-Precision of all algorithms decreases, but the new algorithm maintains its lead all throughout and its e-Precision is at least 6 times higher for all tested concurrency lengths.

Why does the e-Precision of all algorithms decrease when the length of the suspects list is increased? The reason is that the average suspicion level of the accounts in the list decreases with its length. This is because accounts that rank lower in the order computed by the account-level detector are being added to the list, more and more of them coming from communities with smaller spamminess values.

Table II presents more data on the performance of the new algorithm during the first 10 days. For each day d, the respective line lists the following data: 1) the e-Precision for

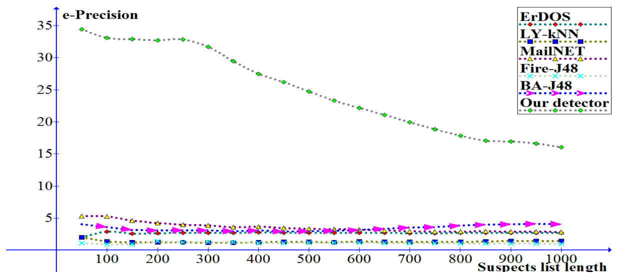

Fig. 7. A comparison of e-Precision values, averaged over 10 days.

TABLE II. OUR DETECTOR'S E-PRECISION AND ENRICHMENT FACTORS FOR THE 100 AND 500 MOST SUSPICIOUS ACCOUNTS.

day	e-Precision (top 100)	e-Precision (top 500)	number of accounts	non-detected spammers	EF (top 100)	EF (top 500)
1	32.0	30.8	1,081,744	5,684	60.9	58.6
2	39.0	29.2	1,384,157	6,887	78.4	58.7
3	35.0	23.2	1,361,968	6,416	74.3	49.2
4	37.0	28.0	1,352,141	6,020	83.1	62.9
5	33.0	20.0	1,300,957	5,661	75.8	45.9
6	31.0	27.0	1,182,576	5,176	70.8	61.7
7	30.0	21.2	846,349	3,573	71.0	50.2
8	29.0	16.6	1,042,037	3,616	83.5	47.8
9	37.0	26.8	1,381,282	4,848	105.4	76.3
10	30.0	15.8	1,345,553	4,399	91.7	48.3

the top 100 and 500 accounts, 2) the total number of internal accounts that have sent/received a message during day d and the number of spammers not yet detected by the content-based filter, and 3) the enrichment factor (see Section VI-A) obtained for the top 100 and 500 accounts.

It can be seen that, on average, 34% (24%) of the top 100 (500) suspects of days 1-10 are early-detected accounts. We note that this is in fact a lower bound on the actual detection rate, since it is plausible that additional listed accounts either send spam that is detected by the content-based detector only at a later period for which we have no data or manage to entirely evade it.

The right-hand side of the table presents enrichment factors for the top 100 and top 500 accounts. We exemplify the calculation of this metric on the top 100 accounts. From the numbers of accounts and non-detected spammers we see that, on average, 0.42% of the allegedly legitimate accounts on days 1-10 send spam that is detected by the content-based filter in later days. Thus, a randomly-chosen list of 100 accounts would include 0.42 early detections, hence the average enrichment factor obtained by our algorithm for the top 100 accounts during the first 10 days is 79.5. The computation of the EF for the top 500 accounts is done similarly.

From looking at the e-Precision figures in Table II, we see that they tend to decrease with time. This follows from the way e-Precision is defined (see Definition 1). In the first days of the period logged by the data set, there is a small number of accounts detected in the past and a large future time-window in which the messages of these accounts can still be detected by the content-based filter. As time passes, the number of spammers already detected by the filter is higher and, more importantly, the future time-window for detection is shorter. Eventually, for the last days, the list mostly includes either spammers all of whose messages evade the filter, or false positives of our detector. Unfortunately, we have no way of distinguishing between the two.

This phenomenon is highlighted by Figure 8 which shows this decrease for lists of various lengths. A similar phenomenon occurs for all algorithms and the new detector maintains its lead during all the period covered by the data set.

Figure 9 presents the precision of our detector for various lengths of suspects list, where each curve represents the averaged statistics over 5 consecutive days. Unlike e-Precision, and although they exhibit some fluctuations, precision results do not exhibit downtrend over time (see Definition 2). This is because, unlike e-Precision, precision depends on the rate of accounts that did not spam on a specific day d, regardless of whether the content-based filter detects them before or after d.

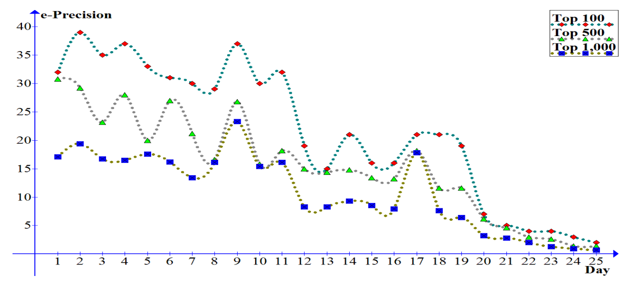

Fig. 8. e-Precision values of our detector over the entire time period, for suspect lists of lengths 100, 500 and 1,000.

Similarly to e-Precision, and for the reason explained above, precision decreases as the length of the suspects list increases.

Figure 10 compares the precision of the evaluated algorithms as a function of the suspects list length, averaged over all the 26 days logged by the data set. Relative performance is similar to that presented for e-Precision on Figure 7. Regardless of list length, the new algorithm significantly outperforms competition by a wide margin.

As we've mentioned in Section III, the data set contains 26 daily files that log outbound email transactions. Emails sent by external email accounts to any of the ESP's internal accounts are recorded in separate *incoming* log files. Unfortunately, our data set contains only 4 daily incoming log files, logging inbound email communication during 4 consecutive days.

The algorithms with which we compare [3], [13], [14], [22], [33] rely on features that can exploit information regarding *incoming* traffic (in addition to outgoing traffic), such as CR (see [22], [33]) and IOR (see [3], [13]). As shown by Figure 7, their performance is quite low when only outgoing logs are available. Next, we compare the performance of the new algorithm, using only outbound logs, with that of the rest of the detectors *when both outbound and inbound logs are available to them*. Clearly, this comparison can only be conducted for the 4 days in which both inbound and outbound traffic were logged. Table III presents the results for a suspects list of length 500 (results are similar for other lengths). It can be seen that, although our detector uses only the outgoing logs

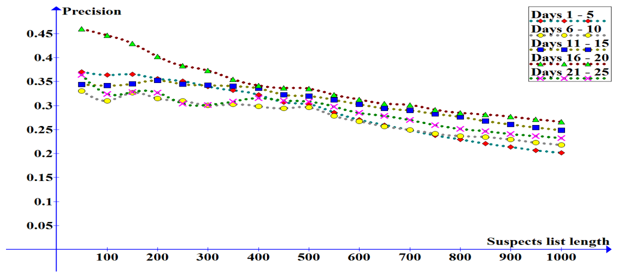

Fig. 9. Precision values of our detector's suspects lists for various lengths.

TABLE III. A COMPARISON OF OUR DETECTOR'S E-PRECISION WITH THESE OF RELATED STUDIES FOR LISTS OF LENGTH 500.

day	Our detector	BA-J48	Fire-J48	ErDOS	LY-kNN	MailNET
1	30.8	11.2	1.4	9.0	1.8	1.8
2	29.2	5.0	2.6	7.6	0.8	1.4
3	23.2	10	2.3	5.2	1.2	2.8
4	28.0	9.9	1.1	6.0	1.2	1.0

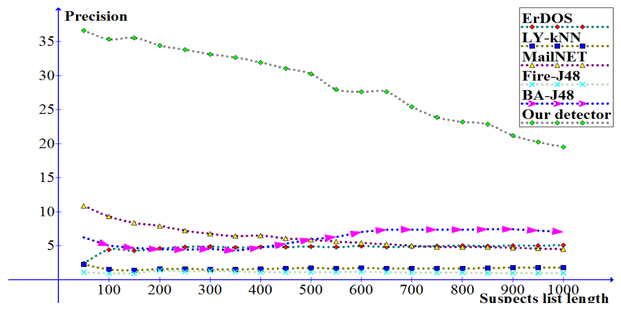

Fig. 10. A comparison of Precision values, averaged over 26 days.

whereas its competitors also use incoming logs, its e-Precision is consistently higher by a factor of between 3.4-28.

VII. Discussion

By using a large real-world data set, provided by a large and well-known ESP, we showed that outgoing spammers tend to be clustered in the same communities. As we explained, this is an unintentional artifact of the manner in which spammers harvest email addresses. We provided an analysis of this phenomenon and identified different types of inter-account communication patterns that give rise to such accidental spammer communities.

Our goal in this work was to achieve early detection of internal spammers, some or all of whose messages evade a content-based filter. We leveraged the existence of spammer communities for devising a highly effective early detection algorithm. Communities with a large fraction of accounts detected by the filter are then input to an account-level detector that assigns a suspicion score to each email account based on its behavior and based on the community to which it belongs. Our experimental evaluation established that the new detector outperforms previous algorithms by a wide margin and identifies a significant fraction of spammers that evade the content-based filter.

On average, early-detected accounts are discovered by our algorithms 10 days before the filter. This is a significant period of time, especially considering that the data-set spans only 26 days. Early-detected accounts send 200 messages during this period on average, which could have been saved by early detection.

References

[1] H.B. Aradhye, G.K. Myers, and J.A. Herson. Image analysis for efficient categorization of image-based spam e-mail. In *ICDAR*, pages 914–918. IEEE Computer Society, 2005.

[2] S. Yousuf B. and Muhammad A. Octracker: A density-based framework for tracking the evolution of overlapping communities in osns. In *ASONAM 2012*, pages 501–505. IEEE Computer Society, 2012.

[3] S. Yousuf B. and Muhammad A. Community-based features for identifying spammers in online social networks. In *ASONAM 2013*, pages 100–107. ACM, 2013.

[4] Vladimir B. and Andrej M. *Pajekanalysis and visualization of large networks*. 2004.

[5] Boldizsár Bencsáth and István Vajda. Efficient directory harvest attacks and countermeasures. *I. J. Network Security*, 5(3):264–273, 2007.

[6] P.O. Boykin and V.P. Roychowdhury. Leveraging social networks to fight spam. *Computer*, 38(4):61–68, 2005.

[7] F. Chung. *Spectral graph theory*, volume 92. AMS, 1997.

[8] F. Moradi et al. An evaluation of community detection algorithms on large-scale email traffic. In *Exp. Alg.*, pages 283–294. 2012.

[9] K. Xu et al. Revealing social networks of spammers through spectral clustering. In *ICC*, pages 1–6. IEEE, 2009.

[10] M. B. Prince et al. Understanding how spammers steal your e-mail address: An analysis of the first six months of data from project honey pot. In *CEAS*, 2005.

[11] S. Venkataraman et al. Exploiting network structure for proactive spam mitigation. In *Proceedings of 16th USENIX Security Symposium*, page 11, 2007.

[12] V. D. Blondel et al. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

[13] Y. Cohen et al. Early detection of outgoing spammers in large-scale service provider networks. In *DIMVA*, pages 83–101. Springer, 2013.

[14] M. Fire, G. Katz, and Y. Elovici. Strangers intrusion detection-detecting spammers and fake proles in social networks based on topology anomalies. *HUMAN*, 1(1):pp–26, 2012.

[15] M. Girvan and M. Newman. Community structure in social and biological networks. *PNAS*, 99(12):7821–7826, 2002.

[16] L.H. Gomes, R.B. Almeida, L. Bettencourt, V. Almeida, and J.M. Almeida. Comparative graph theoretical characterization of networks of spam and legitimate email. *Arxiv preprint physics/0504025*, 2005.

[17] Darya Gudkova. Kaspersky security bulletin. spam evolution 2013. Technical report, Kaspersky, Inc., 2013.

[18] A. Hagberg, P. Swart, and D. S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Laboratory (LANL), 2008.

[19] Oliver Hohlfeld, Thomas Graf, and Florin Ciucu. Longtime behavior of harvesting spam bots. In *IMC*, pages 453–460. ACM, 2012.

[20] J.P. John, A. Moshchuk, S.D. Gribble, and A. Krishnamurthy. Studying spamming botnets using botlab. In *NSDI*, volume 9, pages 291–306, 2009.

[21] N. Krawetz. Anti-honeypot technology. *Security & Privacy, IEEE*, 2(1):76–79, 2004.

[22] H.Y. Lam and D.Y. Yeung. A learning approach to spam detection based on social networks. In *CEAS, 2007*, pages 832–840, 2007.

[23] Fulu Li and Mo-Han Hsieh. An empirical study of clustering behavior of spammers and group-based anti-spam strategies. In *CEAS*, 2006.

[24] F. Moradi, T. Olovsson, and P. Tsigas. Towards modeling legitimate and unsolicited email traffic using social network properties. In *Proceedings of the Fifth Workshop on Social Network Systems*, page 9. ACM, 2012.

[25] Mark EJ Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[26] University of Waikato. Weka 3: Data mining software in Java. http://www.cs.waikato.ac.nz/ml/weka/.

[27] A. Pathak, Y.C. Hu, and Z.M. Mao. Peeking into spammer behavior from a unique vantage point. *LEET*, 8:1–9, 2008.

[28] S. Radicati. Email statistics. Technical report, The Radicati Group, Inc., 2013.

[29] Juan J Rodriguez, Ludmila I Kuncheva, and Carlos J Alonso. Rotation forest: A new classifier ensemble method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1619–1630, 2006.

[30] Jitesh Shetty and Jafar Adibi. The enron email dataset database schema and brief statistical report.

[31] G. Stringhini, O. Hohlfeld, C. Kruegel, and G. Vigna. The harvester, the botmaster, and the spammer: on the relations between the different actors in the spam landscape. In *ASIACCS*, pages 353–364. ACM, 2014.

[32] Bradley Taylor. Sender reputation in a large webmail service. In *CEAS*, volume 27, page 19, 2006.

[33] C.Y. Tseng and M.S. Chen. Incremental SVM model for spam detection on dynamic email social networks. In *CSE*, volume 4, pages 128–135. IEEE, 2009.

[34] B. Viswanath, A. Mislove, M. Cha, and K.P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 37–42. ACM, 2009.