EVILCOHORT: Detecting Communities of Malicious Accounts on Online Services

Gianluca Stringhini[§], Pierre Mourlanne[⋆], Gregoire Jacob[‡],
Manuel Egele[†], Christopher Kruegel[⋆], and Giovanni Vigna[⋆]

[§]University College London [⋆]UC Santa Barbara [‡]Lastline Inc. [†]Boston University
g.stringhini@ucl.ac.uk pmourlanne@gmail.com
gregoire@lastline.com megele@bu.edu {chris,vigna}@cs.ucsb.edu

Abstract

Cybercriminals misuse accounts on online services (e.g., webmails and online social networks) to perform malicious activity, such as spreading malicious content or stealing sensitive information. In this paper, we show that accounts that are accessed by botnets are a popular choice by cybercriminals. Since botnets are composed of a finite number of infected computers, we observe that cybercriminals tend to have their bots connect to multiple online accounts to perform malicious activity.

We present EVILCOHORT, a system that detects online accounts that are accessed by a common set of infected machines. EVILCOHORT only needs the mapping between an online account and an IP address to operate, and can therefore detect malicious accounts on any online service (webmail services, online social networks, storage services) regardless of the type of malicious activity that these accounts perform. Unlike previous work, our system can identify malicious accounts that are controlled by botnets but do not post any malicious content (e.g., spam) on the service. We evaluated EVILCOHORT on multiple online services of different types (a webmail service and four online social networks), and show that it accurately identifies malicious accounts.

1 Introduction

Online services, such as online social networks (OSNs), webmail, and blogs, are frequently abused by cybercriminals. For example, miscreants create fake accounts on popular OSNs or webmail providers and then use these accounts to spread malicious content, such as links pointing to spam pages, malware, or phishing scams [27, 31, 40]. A large fraction of the malicious activity that occurs on online services is driven by *botnets*, networks of compromised computers acting under the control of the same cybercriminal [9].

Leveraging existing services to spread malicious content provides three advantages to the attacker. First, it is easy to reach many victims, since popular online services have many millions of users that are well connected. In traditional email spam operations miscreants have to harvest a large number of victim email addresses (on the web or from infected hosts) before they can start sending spam. On online services such as OSNs, on the other hand, cybercriminals can easily find and contact their victims or leverage existing friends of compromised accounts [15]. In some cases, such as blog and forum spam, cybercriminals do not even have to collect a list of victims, because their malicious content will be shown to anybody who is visiting the web page on which the spam comment is posted [21,31]. A second advantage of using online services to spread malicious content is that while users have become aware of the threats associated with email, they are not as familiar with scams and spam that spreads through other communication channels (such as social networks) [5, 18, 27]. The third advantage is that while online services have good defenses against threats coming from the outside (e.g., emails coming from different domains), they have a much harder time detecting misuse that originates from accounts within the service itself (e.g., emails sent by accounts on the service to other accounts on the same one) [28].

To carry out malicious campaigns via online services, attackers need two resources: *online accounts* and *connection points*. Almost all online services require users to sign up and create accounts before they can access the functionality that these services offer. Accounts allow online services to associate data with users (such as emails, posts, pictures, etc.), and they also serve as a convenient way to regulate and restrict access. Connection points are the means through which attackers access online accounts. They are the devices (hosts) that run the client software (e.g., web browsers or dedicated mobile applications) that allow the miscreants to connect to online services. Often, connection points are malware-infected machines (bots) that serve as a convenient way for the attacker to log into the targeted service and issue the necessary commands to send spam or harvest personal information of legitimate users. However, malicious connection points do not need to be bots. They can also be compromised servers, or even the personal device of a cybercriminal.

In this paper, we propose EVILCOHORT, a novel approach that detects accounts on online services that are controlled by cybercriminals. Our approach is based on the analysis of the interactions between attackers and an online service. More precisely, we look at the interplay between accounts, connection points, and actions. That is, we observe which account carries out what action, and which connection point is responsible for triggering it.

The intuition behind our approach is that cybercriminals use online services differently than regular users. Cybercriminals need to make money, and this often requires operations at a large scale. Thus, when such operations are carried out, they involve many accounts, connection points, and actions. Moreover, accounts and connection points are related in interesting ways that can be leveraged for detection. A key reason for these interesting relationships is the fact that attackers use bots (as connection points) to access the online accounts that participate in an orchestrated campaign. By linking accounts and the connection points that are used to access these accounts, we see that malicious *communities* emerge, and these communities can be detected.

EVILCOHORT works by identifying communities (sets) of online accounts that are all accessed from a number of shared connection points (we use IP addresses to identify these connection points). That is, we observe a number of IP addresses and accounts, and each account is accessed by a non-trivial portion of these IP addresses. Typically, these IP addresses correspond to bot-infected machines, and they are used to log into the accounts that are under the control of the attacker. To identify communities, we consume a log of *interaction events* that the online service records. An interaction event can be any action that a user performs in relation to an account on an online service, such as logging in, sending an email, or making a friend request. Each event also contains the account that is involved, as well as the IP address that sends the request. Our results show that the overwhelming majority of accounts that are identified by our community detection approach are actually malicious, and that therefore the detection by EVILCOHORT is reliable enough on its own. As an additional step to better understand the detected communities and help us assess potential false positives we present techniques to analyze the characteristics of accounts within a community and identify typical behaviors that are indicative of malicious activity. Such characteristics include suspicious activity frequencies over time, synchronized activity of the accounts in the community, and the distribution of the types of browsers used by the infected machines to connect to the online accounts.

One key advantage of our approach is that it is generic, as it does not rely on service-specific information. This is different from previous research, which typically leverages service-specific information to perform detection. For example, BOTGRAPH [39] looks at accounts that are accessed by multiple IP addresses, similarly to our approach, but relies on heuristics based on the email-sending behavior of such accounts to limit false positives. This fact not only makes deployment more problematic, but also limits the applicability of the system to accounts that are misused to send spam. Contrast this with our broad definition of interaction events that is satisfied by a large variety of data that naturally accumulates at online service providers, and makes our approach applicable to any online service that requires users to create an account to interact with it. We demonstrate this by leveraging our approach to detect spammers on a webmail service, as well as to identify malicious accounts on multiple OSNs.

An additional advantage of our approach is that it can be applied to different types of actions. These actions can include account generation and login operations. In these cases, it might be possible to detect malicious accounts before they distribute any malicious content, as an early warning system. Also, it can help to identify abuses where no malicious content is distributed at all. An example of this are botnets that use social networks as part of their command-and-control (C&C) infrastructure [26], or botnets that crawl the online profiles of users harvesting personal information [17]. To show the versatility of our approach, we apply it to two different types of interaction events: on the webmail service we look at events that correspond to the sending of emails, while on the OSNs an interaction event is recorded when a user logs into her account. Over a period of five months, EVILCOHORT detected more than one million online accounts as malicious on the analyzed services. In summary, this paper makes the following contributions:

- We show that a significant amount of malicious activity is carried out by accounts that form communities (when looking at the connection points that access them). We also find that these accounts tend to remain active for extended periods of time on a large webmail provider.
- We present EVILCOHORT, a novel approach to detect malicious communities (and hence, accounts controlled by cybercriminals) on online services. This approach works by detecting accounts that are accessed by a common, shared set of IP addresses.
- We evaluated EVILCOHORT on datasets of different types of interactions collected on five different online services. Over a period of five months, EVIL-

Cohort detected more than one million accounts used to perform malicious activities. We show that EvilCohort is effective in detecting malicious communities regardless of the type of accounts analyzed, making it a valuable tool to protect a variety of online services.

2 Motivation: Analysis of Malicious Activity on a Webmail Service

We want to understand the way in which cybercriminals abuse accounts on online services, to identify weak points that we could leverage for detection. To this end, we observed the email-sending activity on a large webmail service. Our dataset was composed of the emails generated by 21,387,006 distinct online accounts over a period of one day. In total, this dataset contained 72,471,992 emails. We call the dataset containing information about this email-sending activity **T**. For each email-sending event, the dataset **T** contains the IP address that accessed the account, the user ID of the account that sent the email, and a timestamp. In addition, each email-sending event contains information on whether the email was considered as spam by the webmail provider or not. Note that the dataset **T** only contains information about sent emails, and provides no insights on the number of times an account is accessed without sending any email (e.g., to check the account's inbox).

Two Types of Malicious Accounts. We analyzed the accounts that sent spam in the dataset **T**. We identify two types of malicious accounts:

1. *Accounts that are used in isolation*. Each account is accessed by a single IP address, which could be the attacker's computer or a single infected machine.
2. *Accounts that are accessed by multiple IP addresses*. The same account is accessed by multiple infected computers.

We looked at how many malicious accounts of each type are active on the webmail service. For this analysis we considered an account as malicious if the account sent at least 10 emails during the day under consideration, and the majority of these emails were flagged as spam by the webmail provider. We selected this threshold because we needed a set of "labeled" accounts that sent spam on the webmail provider. Picking accounts whose majority of emails was flagged as spam by the email provider gives us confidence that this dataset does not contain false positives. Note that this preliminary analysis was purely qualitative, and it was used to give us an idea on the behavior of malicious accounts on a webmail service. We call this set of labeled accounts **L**. In total, **L** is composed of 66,509 malicious accounts that were accessed by a single IP address, and 103,918 malicious accounts that were accessed by two or more.

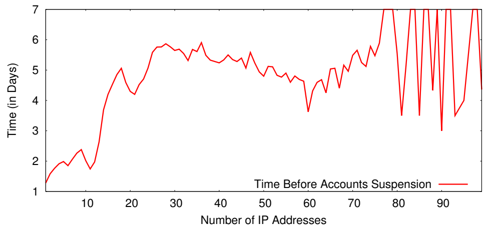

Figure 1: Average time (in days) before a spamming account was suspended in **L**, given the number of IP addresses accessing that account.

Accounts Shared by Many IP Addresses Are More Dangerous. We then investigated the effectiveness of the two identified types of spam accounts in sending emails, and their ability to evade detection by the webmail provider. With detection, we mean triggering a mechanism on the webmail provider that leads to the account being suspended. Figure 1 shows the average time (in days) that it took for a malicious account in **L** to be suspended after it sent the first spam email, given the number of IP addresses that accessed that account. As it can be seen, accounts that are used in isolation have a shorter lifespan than the ones that are used by multiple IP addresses: accounts that are only accessed by a single IP address are typically detected and suspended within a day, while ones that are accessed by many different IPs can survive for as long as a week.

We then studied the difference in the activity of the two types of accounts with regards to the number of spam emails sent. Figure 2 shows that accounts that are used in isolation are less effective for cybercriminals, as they send a smaller number of emails per day before being shut down. Alternatively, attackers can have each of their infected computers send a small number of emails and stay under the radar. Figure 3 shows that IP addresses accessing accounts used in isolation send 19 emails per day on average before being blocked, while having multiple computers accessing the same account allows cybercriminals to have each IP address send a lower number of emails, as low as one email per IP address in some cases. The longevity of the accounts that are accessed by more than one IP address suggests that the webmail service lacks effective countermeasures to prevent abuse of the service by such accounts. We acknowledge that this could be due to shortcomings in the countermeasures deployed by this particular webmail service, but it still shows us that accounts that are accessed by a multitude of infected computers are a problem for online services.

Detecting Malicious Accounts Shared by Many IP Addresses. Can we use the fact that malicious accounts tend to be accessed by many IP addresses to flag these

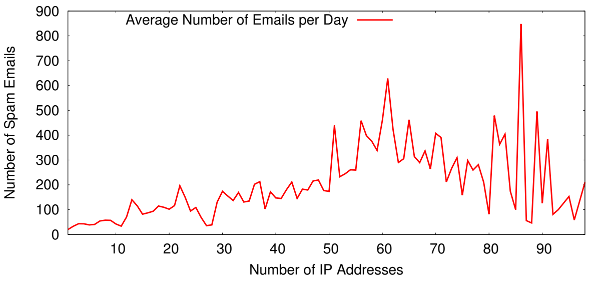

Figure 2: Average number of spam emails sent per day per *account* accessed by a certain number of IP addresses.

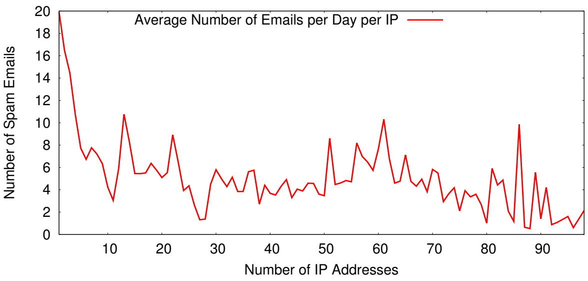

Figure 3: Average number of spam emails sent per day per *IP address* that accessed a certain account.

accounts as malicious? Unfortunately, the number of IP addresses that accessed an account is not a strong enough indicator and basing a detection system only on this element would generate a number of false positives that is too high for most practical purposes. For example, considering as malicious accounts that were accessed by two or more IP addresses in **T** would cause 77% of the total detections to be false positives (i.e., accounts that did not send any spam email). This makes sense, because many users access their webmail account from different devices, such as a mobile phone and a desktop computer. Even looking at accounts accessed by a higher number of IP addresses does not solve the false positive problem: looking at accounts that were accessed by ten or more distinct IP addresses in **T** 32% would be labeled malicious incorrectly (i.e., false positives); by increasing the number of required IP addresses false positives decrease, but they remain well above the level considered acceptable in a production environment.

To overcome the false positive problem, we leverage another property of cybercriminal operations that use online services: cybercriminals can only count on a limited number of infected machines (bots) [26], as well as a limited number of accounts on the online service. Because of this limitation, and to make their operations more resilient to takedowns, cybercriminals have multiple bots connect to the same set of accounts over time. We can think of a set of accounts that are accessed by the same set of bots as a *community*. In the following, we present EVILCOHORT, a system that detects communities of accounts that are accessed by a common set of IP addresses. We show that, by looking at these communities of accounts, we can detect most of the malicious accounts that are accessed by multiple IP addresses, while generating a false positive rate that is orders of magnitude lower than just looking at accounts in isolation. In Appendix 5.2, we compare the two methods in detail, and show that EVILCOHORT outperforms the method that looks at individual accounts only.

3 EVILCOHORT: Overview

EVILCOHORT operates on inputs in the form of *account interaction events*. Users create their own accounts and connect to online services to perform a number of actions. Depending on the service, these actions range from sending messages to the user's friends and colleagues, to performing friend requests, to browsing pictures, to updating the user's profile. Accounts allow the online service to attribute any activity performed to a specific user, in a more precise way than source IP addresses do. For instance, it is possible to correctly attribute the activity of a certain user regardless of the place she is connecting from (her home computer, her office, or her mobile phone). We define a user interaction with an online service as a tuple

$$\mathbf{A} = < H, U, T >,$$

where H is the host that the user is connecting from (identified by an IP address), U is her user ID on the online service, and T is a timestamp.

Approach Overview. EVILCOHORT works in three phases. First, it collects interaction events from the monitored online service, and builds a bipartite graph where one set of vertices is the online accounts observed and the other set of vertices is the list of IP addresses that accessed them. Then, it computes the weighted one-mode projection of the bipartite graph onto the account vertex set. The result of this phase is a graph, which we call *projected graph representation*, in which the vertices are the accounts and the edge labels (i.e., weights) indicate how many shared IP addresses connected to each pair of accounts. As a third phase, EVILCOHORT performs clustering on the projected graph representation to find communities of online accounts that were accessed by a common set of IP addresses. A last, optional step consists of analyzing the discovered communities, to characterize them and possibly identify security relevant activity, such as campaigns. In the remainder of this section, we provide more details about the three steps involved in identifying communities.

3.1 Data Collection

In the first phase, EVILCOHORT collects interaction events on an online service for a given observation period (a day in our current implementation). Based on these interaction events, EVILCOHORT builds a bipartite graph where the first set of vertices **A** are the online accounts that generated the events, while the second set of vertices **I** are the IP addresses that accessed these accounts. An account vertex has an edge to an IP address vertex if that account was accessed by that IP address. We call this bipartite graph $\mathbf{G}_A$.

3.2 Building the Projected Graph Representation

We expect that cybercriminals instruct their bots to connect to multiple accounts under their control. As discussed in Section 2, this is because they have control of a limited number of bots and want to optimize the effectiveness of their malicious operation. For this reason, we represent the relation between online accounts and IP addresses as a weighted graph. To this end, we perform the weighted one-mode projection of the bipartite graph $\mathbf{G}_A$ onto the account vertex set **A**. More precisely, we define the *projected graph representation* of the set of accounts **A** as

$$\mathbf{R} =< \mathbf{V}, \mathbf{E} >,$$

where each element in the set of vertices **V** is one of the accounts in **A**, and the set of edges **E** is weighted as follows: for each pair of accounts u_1, $u_2 \in \mathbf{V}$, the edge connecting them has a weight equal to the number of IP addresses that u_1 and u_2 share, based on the bipartite graph $\mathbf{G}_A$. If the accounts u_1 and u_2 do not share any IP address, there is no edge between them.

As we showed in Section 2, many legitimate accounts are accessed by more than one IP address. To focus on detecting communities of accounts that share a higher number of IP addresses, we filter the bipartite graph $\mathbf{G}_A$ on the in-degree of the accounts in **A**. More precisely, we introduce a threshold s, and consider as inputs for the projection only those accounts that have a degree higher than s, which means that they were accessed by more than s IP addresses during the observation period. Since the number of IP addresses that legitimate accounts share is low, communities of accounts sharing many IP addresses are suspicious. We investigate the possible choices for the threshold s in Section 5.1. By increasing the value of s we can reduce false positive considerably, but we also reduce the number of accounts that EVILCOHORT can detect as malicious. The graph **R** is then passed to the next phase of our approach, which finds communities of online accounts that are accessed by a common set of IP addresses.

3.3 Finding Communities

After obtaining the projected graph representation **R**, we identify communities of accounts. To this end, we use the "*Louvain Method*" [6]. This clustering method leverages an iterative algorithm based on modularity optimization, and is particularly well-suited to operate on sparse graphs, as most graphs obtained from "real life" situations are [12]. In their paper, Blondel et al. [6] show that their method outperforms several community-detection algorithms that are based on heuristics.

The Louvain method operates in two steps, which are iteratively repeated until convergence is reached. At the beginning, each vertex in **R** is assigned to its own community of size one. Each iteration of the algorithm proceeds as follows:

1. For each account u_1 in **U**, we consider each of its neighbors u_2, and we calculate a gain value g that represents the effect of removing u_1 from its community and adding it to u_2's community. We explain how we calculate g later in this section. If any of the gain values g is positive, we move u_1 to the community of the account that yields the highest gain.
2. We rebuild the graph **R**, whose nodes are now the communities built during the previous step. Each edge between two communities c_1 and c_2 is weighted with the number of IP addresses that are shared between the two communities.

The algorithm repeats these two steps until convergence. Blondel et al. [6] describe how the gain value g is calculated in detail. In a nutshell, the gain obtained by moving an account i to a community C is

$$g_{in} = [\frac{\sum_{in} + k_{i,in}}{2m} - (\frac{\sum_{tot} + k_i}{2m})^2] - [\frac{\sum_{in}}{2m} - (\frac{\sum_{tot}}{2m})^2 - (\frac{k_i}{2m})^2],$$

where $\sum_{in}$ is the sum of the weights of the edges between the accounts in C, $\sum_{tot}$ is the sum of the weights of the edges incident to the accounts in C, k_i is the sum of the weights of the edges incident to i, $k_{i,in}$ is the sum of the weights of the edges that connect i to the accounts in C, and m is the number of edges in **R**. Blondel et al. show how a similar weight is calculated for the gain obtained by removing an account i from its community (g_{out}) [6]. If the sum of the two gains g = $g_{in} + g_{out}$ is positive, the account i gets added to the community C.

3.4 Optional Step: Characterizing Communities

As an optional step, after the detection of (malicious) communities, we propose a number of techniques that extract interesting properties of these communities. These properties allow the operator of EVILCOHORT to easily characterize security-relevant behaviors of the communities. As we will see later, these properties can be useful to both assess false positives and identify whether the accounts in a community are fake (i.e., Sybils) or compromised accounts that are accessed both by cybercriminals and by their legitimate owners. To gain insight into the behavior of accounts associated with communities these properties can incorporate auxiliary sources of information that are not strictly part of the collected *account interaction events* (e.g., web-browser user agents).

User agent correlation. Regular users of online services likely connect to their accounts from a limited set of devices corresponding to a largely consistent set of connection points. During the course of a day, a typical user would access, for example, her account from home using her personal browser, then from work using the browser mandated by company policy, and finally from her phone using her mobile client. In other words, we expect to have a one-to-one relation between the connection points and the client programs (agents) that run on these machines and are used to perform the activity. When online services are accessed via the web, the client used to perform the activity can be identified by the HTTP user agent field. Proprietary clients often have similar attributes that can be used for this purpose. On iOS, for example, the system-provided HTTP library uses the application's name and version as the user agent string.

For malicious communities, the activity is no longer generated by humans operating a browser. Instead, the activity is frequently generated by autonomous programs, such as bots, or programs used to administer multiple accounts at once. These programs can be designed to use either hard-coded user agent strings, or, as we observed in recent malware, slight variations of legitimate user agent strings. Presumably, this is a technique aimed at evading detection mechanisms. However, these practices significantly change the distribution of user agent strings and their corresponding connection points.

To measure the correlation between connection points and user agents within a community c, we compute the following ratio:

$$log(c) = log\left(\frac{\text{number of user agents}}{\text{number of IP addresses}}\right) \qquad (1)$$

For a typical benign user, the correlation is very strong because there is a one-to-one relationship between connection point and user agent: That is, each connection point is associated with a different user agent, and as a result $log(c)$ tends towards 0. For malicious communities, where the relationship becomes one-to-n, negative values will be observed in case of hard-coded user agent strings, and positive values in case of permutations of the user agent strings. Note that we exclude from the computation user agent strings coming from mobile phones or tablets because these mobile devices can be connected to any network, meaning that no correlation can be expected in this case.

Event-based time series. This property captures *account interaction events* in a community over time. Time series represent the frequency of events per time period. As we will show in Section 6, the time series representations fundamentally differ for legitimate accounts and those in malicious communities. Time series for legitimate users commonly contain daily activity patterns depending on the night and day cycle. Furthermore, weekly patterns can often be identified too. Automated malicious activity, however, commonly results in either highly regular activity (e.g., botnets using the online service as their command and control service), or irregular bursts (e.g., during the execution of a spam campaign).

IP address and account usage. This analysis, similarly to the previous one, relies on timing analysis. The main difference is that events are no longer aggregated for the entire community but, instead, individual IP addresses and accounts are represented separately over time.

The IP addresses usage graph is generated in the following way: Time is represented on the x-axis, and each unique IP address is represented by a separate entry on the y-axis of the graph. Events are then plotted as points in the graph using this set of coordinates. The account usage graph is generated in a similar way, with unique accounts instead of IPs on the y-axis. We will show an example for IP address and account usage graphs in Section 6. Similar to the above time series representation, malicious communities show a high degree of synchronization, which is not present for communities formed by legitimate users. This observation has been confirmed by independent research that has been recently published [7]. Using this type of representation, any suspicious alignment in the events recorded for different IP addresses or different accounts can easily be identified.

Automated post-processing. In our current implementation we exclusively use the analysis of community properties to infer interesting characteristics of identified communities. Our analysis indicates that communities formed by malicious accounts exhibit vastly different characteristics than those formed by legitimate accounts, as we show in Section 6. Thus, the techniques described in this section could also be used to automatically distinguish malicious from legitimate communities. Similar to Jacob et al. [17], detection could be implemented based on automated classifiers working on statistical features characterizing the shape of a time series or plot. While such an automated post-processing approach would be a potential avenue for reducing the false positives of EVILCOHORT even further, our current false positives are already well within the range where a deployment would benefit an online service. An implementation of this automated post-processing step is, thus, left for future work.

4 Description of the Datasets

In Section 2, we analyzed **T**, a labeled dataset of email-sending events on an webmail provider. Since EvilCohort only takes into account the mapping between IP address and online account of an event, however, it can operate on any online service that allows users to create accounts. Such services include web-based email services, online social networks, blogs, forums, and many others. In addition, EvilCohort can operate on activities of different type, such as login events, message postings, message shares, etc. To show the versatility of our approach, we evaluated it on multiple datasets of activities on five different online services. The first dataset is composed of email sending events logged by a large webmail service, with similar characteristics to **T**. The second dataset is composed of login events recorded on four different online social networks. In the following, we describe these datasets in more detail.

4.1 Webmail Activity Dataset

Our first dataset is composed of email-sending events logged by a large webmail provider. Every time an email is sent, an activity is logged. We call this dataset $\mathbf{D}_1$. Note that the email-sending events in this dataset are generated by accounts on the webmail service, which send emails to other email addresses.

The dataset $\mathbf{D}_1$ contains the events logged over a five-month period by the webmail provider for a subset of the accounts that were active on the service during that period. In total, this dataset contains 1.2 billion email-sending events, generated by an average of 25 million accounts per day. This data was collected according to the webmail provider's terms of service, and was only accessed on their premises by a company's employee. Beyond the above-discussed information, no further email related information was accessible to our research team (i.e., no content, no recipients). In addition to the activity events, the webmail provider logged whether the email was flagged as spam by their anti-spam systems. The dataset **T** presented in Section 2 is a subset of $\mathbf{D}_1$. We used **T** to study in-depth the properties of legitimate and malicious accounts on a webmail service (see Section 2). As we explained in Section 2, **T** is a dataset containing email events observed on a large webmail provider over a period of one day, while **L** contains the accounts in **T** that are heavy senders of spam (meaning that they sent 10 or more emails during the day of observation, and that a majority of these emails was detected as spam by the defenses in place at the webmail provider).

It is worth noting that **L** does not contain all the accounts that sent spam in **T**. Such ground truth does not exist, because if a perfect detection system existed we would not need new approaches such as EvilCohort. Instead, **L** contains a set of "vetted" spam accounts that were detected by the webmail provider, and using this dataset as a reference allows us to get a good idea of how well EvilCohort works in detecting previously-unseen malicious accounts on online services.

4.2 Online Social Network Login Dataset

Online Social Network	**OSN$_1$**	**OSN$_2$**	**OSN$_3$**	**OSN$_4$**
Login events	14,077,316	311,144	83,128	42,655
Unique Accounts	6,491,452	16,056	25,090	21,066
Unique IPs	6,263,419	17,915	11,736	4,725
Avg. daily events	2,067,486	51,832	11,897	6,601
Account singletons	74.6%	40.0%	51.7%	72.2%

Table 1: Statistics of activity events of the dataset $\mathbf{D}_2$.

Our second dataset is composed of login events collected from four different OSNs, spanning a period of 8 days. We call this dataset $\mathbf{D}_2$. We obtained the dataset $\mathbf{D}_2$ from a security company. For each activity event, the dataset contained additional information such as the user agent of the web browser performing the login and the HTTP headers of the response. Sensitive information such as the user IDs and the IP addresses was anonymized. Note that this does not affect our community detection algorithm at all.

Statistics on the number of login events for each social network can be found in Table 1. These statistics reflect the size and activity observed on these networks, ranging from tens of thousands up to 14 million login events. One interesting observation is the high percentage of account singletons on a daily basis, i.e., the percentage of users connecting at most once a day. On a weekly basis, the percentage tends to drop but remain surprisingly high. These users are probably legitimate users that are not very active on the social network.

5 Evaluation

In this section, we analyze how EvilCohort performs in the real world. We first study the effectiveness of our approach by using the dataset **T** and its subset **L** of labeled malicious accounts. We then select a suitable threshold s that allows us to have a small number of false positives. Finally, we run EvilCohort on multiple real-world datasets, and we analyze the communities of malicious accounts that we detected.

Value of s	# of accounts	# of communities	Known accounts in L	Additional detections over L	FP communities	FP accounts
			(% of tot. accounts in L)	(% of add. detections over L)	(% of tot. communities)	(% of tot. accounts)
2	135,602	3,133	94,874 (58%)	40,728 (23.8%)	1,327 (42%)	12,350 (9.1%)
5	77,910	1,291	51,868 (30.4%)	26,042 (15.2%)	580 (44.9%)	2,337 (3%)
10	25,490	116	16,626 (9.7%)	8,864 (5.2%)	48 (41.3%)	433 (1.7%)
65	1,331	6	1,247 (0.7%)	84 (0.04%)	0	0

Table 2: Summary of the results reported by EvilCohort for different values of the threshold s.

5.1 In-degree Threshold Selection

As with every detection system, EvilCohort has to make a trade-off between false negatives and false positives. As we mentioned in Section 3.2, we can adjust the value of the minimum in-degree for account vertices that we use to generate the one-mode projection graph **R** to influence the quality of EvilCohort's results. We call this threshold s. In particular, increasing the value of s decreases the number of false positives of our system, but also reduces the number of accounts that can be detected. That is, any account that is accessed by less than s IP addresses during an observation period is excluded from evaluation for community membership, and thus cannot be detected as malicious by EvilCohort.

In this section we run EvilCohort on the datasets **T** and **L** and analyze the quality of its results. The goal is to identify a suitable value of s for running EvilCohort in the wild. Recall that **L** is the set of accounts that were classified as malicious as explained in Section 2. In the absence of complete ground-truth, we use **L** as a partial ground-truth to help us assess how well EvilCohort operates.

The first element that we use to evaluate the effectiveness of EvilCohort is the fraction of accounts in **L** that our system is able to detect. Ideally, we want EvilCohort to detect a large fraction of our labeled malicious accounts. Unfortunately, as discussed above, increasing the value of s decreases the number of accounts that EvilCohort can possibly detect. The percentage of malicious accounts in **L** detected by EvilCohort provides us with an estimate of the false negatives that EvilCohort would report if it was run in the wild.

As a second element of effectiveness, we look at the set of accounts that EvilCohort detects as malicious in **T**, but that were missed by the anti-spam systems deployed by the webmail provider. These are malicious accounts *not* in **L**. We refer to this number as *additional detections*. This value gives us an estimate on the overall effectiveness of EvilCohort. Ideally, we want this number to be high, so that if EvilCohort were to be deployed in conjunction with the defenses that are already in place on the online service, it would increase the number of malicious accounts that can be detected and blocked.

The third element that we consider is the *confidence* that the communities detected by EvilCohort are indeed malicious. To this end, we look at the fraction of accounts in **L** that are present in each detected community. We consider a community as malicious (i.e., a true positive) if at least 10% of the accounts belonging to it are part of our labeled dataset of malicious accounts. Otherwise, we consider it as a false positive of EvilCohort. We empirically found that this 10% fraction of vetted bad accounts gives us a good confidence that the communities are indeed malicious. Recall that **L** is a dataset composed of "repeated offenders." In other words it contains accounts that have a consistent history of sending spam, therefore having a small fraction of accounts from this set in a community is a strong indicator of the entire community being malicious. As we show in Section 5.3, if we relax the method that we use to assess true positives (for example we consider an account as malicious if it sent a single email flagged as spam by the webmail provider) then the majority of the accounts in communities detected by EvilCohort are confirmed as malicious. In Section 6 we show that by observing additional properties of the communities detected by EvilCohort we are able to confirm almost the totality of them as malicious.

Table 2 provides a summary of the results that we obtained when running EvilCohort on **T**, based on different values of the threshold s. As one can see, the fraction of accounts in **L** that our system detects decreases quickly as we increase s. With a threshold of 2, EvilCohort only detects 58% of the labeled accounts.With a threshold of 10 the fraction of accounts in **L** that are covered is only 10%. Once we reach higher thresholds, the fraction of detected accounts that are part of **L** becomes very small. The additional detections performed by EvilCohort over the webmail provider's detection system also decrease as we increase s. With a threshold of 2 we detect 23% malicious accounts that existing approaches miss. A threshold of 10 still ensures 5.5% additional detections over the dataset **L**. False positives decrease rapidly as we increase s as well. Setting s to 2 results in 9% false positives. A threshold of 10 reduces false positives to 1.7%. By setting s to 65, EvilCohort does not mistakenly flag any legitimate account as malicious. Unfortunately, the number of detections at this threshold is quite low.

Given the results reported in this section, we decided to use 10 as a value of s for our experiments. At this threshold false positives are low (1.7%), but the system is still able to significantly improve the detections performed by the existing countermeasures deployed by the webmail provider, and detects 116 communities.

It is interesting to notice that although the number of accounts misclassified by EVILCOHORT is generally low, percentages are higher when looking at communities: with a threshold of 10, for example, 40% of the detected communities are considered to be false positive accounts. Interestingly, however, the size of true positive and false positive communities varies consistently: false positive communities are composed of 9 accounts on average, while malicious ones are composed of 370 or more accounts. An explanation for this is that small communities could be a side effect of computers behind a NAT which constantly change their IP address through DHCP. As previous research showed, some ISPs change the IP address of their customers very often [26]. The small size of such communities, however, shows that filtering on the number of accounts in a community could be an effective filter to further reduce false positives. We did not include a threshold on the size of a community in EVILCOHORT because we wanted to keep the system general. However, in a production setting an additional threshold on the community size could be a straight-forward way to reduce false positives even further. In addition, Section 6 illustrates that false positive communities expose distinct behavioral characteristics. These characteristics can be leveraged to further reduce false positives.

5.2 Comparison between EVILCOHORT and the Single Account Method.

As we discussed in Section 2, EVILCOHORT outperforms detection approaches that look at single accounts accessed by a high number of IP addresses by orders of magnitude. At a threshold of 10, where EVILCOHORT reports a false positive rate of 1.7%, the single-account method has a false positive rate of 32%. Even by dramatically increasing the threshold, the number of false positives of the single-account method remains high. At a threshold of 65, at which EVILCOHORT reports no wrong detections, the single-account method has a false positive rate of 1.2%. Even at a threshold of 100, the single-account method has a small number of false positives.

The last question to answer is whether accounts that are accessed by a high number of IP addresses do form communities, in other words whether EVILCOHORT is able to detect most malicious accounts that were accessed by a number of IP addresses s. To answer this question, we looked at single accounts accessed by a number of IP addresses n (from one to 100), and labeled them as malicious or benign in the same way we labelled the communities in the previous experiment. We then proceeded as follows: for each value of n, we considered the single-account method to have perfect recall (i.e., no false negatives). We then looked at how many of the accounts detected by this method would have formed communities, and therefore be detected by EVILCOHORT. The fraction of malicious accounts that form communities is generally very high. With a threshold of 10, EVILCOHORT detected 93% of the malicious accounts detected by the single-account method. With a threshold of 20 this fraction becomes 95%, while with a threshold of 50 it becomes 98%. We conclude that the vast majority of accounts accessed by a high number of IP addresses form communities, and that therefore EVILCOHORT is a suitable alternative to the single-account method for what concerns false negatives, and it reduces false positives by orders of magnitude compared to the single account method.

5.3 Detection in the Wild

We applied EVILCOHORT to the datasets $\mathbf{D}_1$ and $\mathbf{D}_2$. In the following, we show that EVILCOHORT is able to detect a large number of malicious online service accounts, regardless of the type of online service that it is run on.

Detection on the webmail activity dataset. The dataset $\mathbf{D}_1$ is composed of email-sending activities logged on a large webmail provider. Over a period of 5 months, EVILCOHORT detected **M** = 1,217,830 accounts as malicious. In total, these accounts were part of 17,803 malicious communities.

We first wanted to understand the number of false positives generated by EVILCOHORT in the wild. We tried to answer this question from two vantage points. First, we performed the same false positive analysis explained in Section 5.1. That is, we considered an account to be vetted as malicious if the majority of the emails sent by it during the day of observation were detected as spam by the webmail operator. We then considered a community as a false positive by EVILCOHORT if less than 10% of the accounts in the community belonged to our set of vetted malicious accounts. In total, we found 23,269 accounts to be potential false positives (1.9% of the total accounts in **M**). This is in line with the validation results from Section 5.1, in which we reported 1.7% false positives by using the same threshold. As a second method of assessment, we manually analyzed 100 randomly picked communities among the ones detected by EVILCOHORT. For each of these communities we could identify signs of automated activity and possible maliciousness. For example, the user IDs of the accounts used by some communities had been clearly automatically generated: in one case, all the accounts belonging to the community were composed of two dictionary words concatenated with a four-digit number. In another case, all the user IDs were 20-letter random alphanumeric characters. In an-

Day	1	2	3	4	5	6	7	8
OSN_1	24	30	6	4	4	4	5	2
OSN_2	1	0	0	0	0	0	0	0
OSN_3	0	0	0	0	1	1	1	0
OSN_4	0	0	0	0	0	0	0	0

Table 3: Number of malicious communities detected per day by EvilCohort on the dataset $\mathbf{D}_2$.

other case, the accounts belonging to a community were accounts with a long history of legitimate activity, which suddenly started being accessed by a large, common set of IP addresses. We highly suspect that this community of accounts was composed of legitimate accounts that had been compromised by cybercriminals.

We then wanted to evaluate the false negatives reported by EvilCohort. 94.6% of the vetted malicious accounts used in the false positive analysis at the operating threshold formed communities, and were therefore detected by EvilCohort. The false negatives would therefore account for 5.4% of the total accounts in $\mathbf{M}$. This shows that malicious accounts accessed by a large number of IP addresses are typically accessed by botnets, and confirms the usefulness of our approach.

We then looked at how many accounts detected by EvilCohort sent at least one email that was flagged as spam by the webmail provider during the day in which EvilCohort detected them. In total, 715,671 accounts fall in this category (i.e., 59% of $\mathbf{M}$). This also shows that relaxing our ground truth assumptions and looking at accounts that sent a single spam email as malicious instead of a vetted dataset results in having the majority of the accounts detected by EvilCohort confirmed by the defenses already in place at the webmail provider. Conversely, EvilCohort proves to be able to grow the set of malicious accounts detected by the webmail provider consistently, since it detected 502,159 additional accounts as malicious (41% of the total).

On average, our prototype implementation of EvilCohort processes one day of data in about ten minutes using a COTS server with 16GB of RAM.

Detection on the social network login dataset. The dataset $\mathbf{D}_2$ is composed of login events from online social networks. In the following, we applied EvilCohort to this second dataset to demonstrate the viability of the approach for different types of services. We used the same threshold as selected in Section 5.1 on this dataset too. Unfortunately, no labeled dataset was available for these experiments. To confirm that the accounts belonging to the identified communities were indeed malicious, we performed a post-processing analysis. We discuss the results of these experiments in Section 6.

Social Network	OSN_1	OSN_2	OSN_3	OSN_4
Accounts (Avg)	3,662	2	2	0
Accounts (Med)	13	2	2	0
Accounts (Max)	66,764	2	2	0
IPs (Avg)	2,381	14	10	0
IPs (Med)	19	14	10	0
IPs (Max)	3,9884	14	10	0

Table 4: Size of the malicious communities detected by EvilCohort on the dataset $\mathbf{D}_2$. Numbers (Average, Median and Maximum) are expressed per community.

Over eight days, EvilCohort was able to detect a total of 83 communities, which represents a total of 111,647 unique accounts. The number of detected communities and the size of these communities evolve daily, as one can see in Table 3 and Table 4. Unsurprisingly, our numbers heavily depend on the size of the social network. OSN_1 is by far the largest network; consequently, this is where we observed the highest number of communities, as well as the largest communities. Interestingly, we observe an important drop in the number of communities on the third day. This might indicate that accounts were taken down by the network. The remaining communities tend to be of smaller size. In OSN_2 and OSN_3, we only detect isolated communities of very small size: two accounts accessed by ten different IP addresses. The activity for OSN_4 was too little to detect any interesting community with the selected threshold.

To understand the evolution of the detected communities, we studied their similarity over time. A community remains stable over time if it is found similar to a community detected the day before. We qualitatively consider two communities as similar if they share more than 50% of their accounts. In OSN_1, one of the largest communities, with more than 66,000 accounts, was stable over five days, with a similarity ranging between 53% to 85%. Two communities of smaller size, with about 10,000 accounts each, were found stable over the two first days but they disappeared on the third day as previously observed. The community detected in OSN_3 is only made up of two accounts and it is stable over three days before being brought down.

6 Application of Post-processing Techniques

As we mentioned in Section 2, EvilCohort was motivated by observations performed on a webmail service. Given the generality of our approach, however, we can apply it to any online service that makes use of accounts. For this reason, we tested EvilCohort on the OSN dataset $\mathbf{D}_2$. The question remains on how well EvilCohort works on such a different dataset. Unfortunately, the dataset $\mathbf{D}_2$ came without ground truth. For this reason, we used the techniques described in Section 3.4 to assess the maliciousness of the detected communities. In a nutshell, we analyze the communities detected by

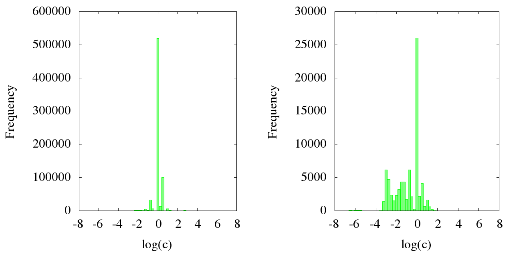

Figure 4: Correlation between user agents and IPs: legitimate accounts (left) and malicious accounts (right).

EvilCohort on the dataset $\mathbf{D}_2$, and consider them as true positives if the accounts belonging to it show patterns that are indicative of automated activity or of botnet use. These techniques are not part of EvilCohort's core detection, but as we will show they are helpful in assessing how well our approach performs.

The results show that most communities of accounts detected by EvilCohort show very different characteristics than legitimate accounts, and are, therefore, very likely malicious. Specifically, of the 83 malicious communities detected by EvilCohort on $\mathbf{D}_2$ only 5 showed characteristics that are similar to regular accounts and are therefore possibly false positives. In particular, we observed a very large community of 5,727 accounts, and four smaller ones of 7, 3, 3, and 2 accounts respectively. Given the size of the largest community, we wanted to understand whether it was really a false positive. Looking at its characteristics, we observed a mix of IP address accessing a single account and a large population of IP addresses accessing many different accounts, which makes us believe that such accounts might have been compromised and being accessed by both their legitimate owners and the hijackers. As such, this community is not a false positive, as it was actually accessed by a botnet. We provide further evidence that this is the case in the following sections. The other communities, on the other hand, are very likely to be false positives by EvilCohort, because they show a consistent human like behavior. Given their small size (15 accounts in total, out of 111,647 total detected accounts), however, we can conclude that the false positives generated by EvilCohort on the dataset $\mathbf{D}_2$ are minimal. In the following, we describe our analysis in detail.

User-agent correlation. Information about user agents was only available for OSN_1 and OSN_2 in $\mathbf{D}_2$. Consequently, we excluded OSN_3 and OSN_4 from this analysis. We also excluded all the account singletons, because the notion of ratio then becomes meaningless.

Based on the description in Section 3.4, we plot the correlation between user agents and IP addresses in Figure 4. The left distribution is generated for legitimate accounts that do not form communities, whereas the right distribution corresponds to accounts in identified malicious communities. The distribution for malicious communities is shifted and no longer aligned on the origin. For legitimate accounts, the average of $log(c)$ was 0.08, which is close to zero, as expected (with a standard deviation of 0.43). For malicious communities, the average shifts to -0.85 with a standard deviation of 1.24.

For the accounts in two of the potential false positive communities described before, the correlation index $log(c)$ was very close to zero, making them very similar to what is expected for regular accounts. For the remaining potential false positive communities this metric did not reveal any anomalous behavior.

Event-based time series. Time series become only significant if the amount of data is sufficiently large to make a measure statistically meaningful. For this reason, we only computed the event-based time series (introduced in Section 3.4) for OSN_1. Unfortunately, the volume of login events observed for OSN_2, OSN_3 and OSN_4 made this approach impractical for these networks.

The assumption behind the time series analysis is that part of the events observed in malicious communities are the result of automation. This results in a distinct shape of activity from communities of legitimate users where events are triggered by humans [17]. To verify this assumption, we plotted the time series associated with the activity of the biggest malicious communities detected in OSN_1. The experiments show that the time series generated for malicious communities differ fundamentally in shape from regular user activity, even when users are grouped behind a NAT.

Concrete examples are plotted in Figure 5 The left time series represents the activity of all users from OSN_1 over 8 days. The reader can clearly see the daily patterns in the activity. The middle time series represents the activity generated by the largest community detected in OSN_1. As can be seen, there are fundamental differences: disappearance of the daily patterns and higher stability on the long term. The right time series is representative of most of the time series obtained for smaller communities of OSN_1: the volume of events remains low but one can clearly observe regular bursts of activity. This bursty shape is also observed for the potential false positive community of 5,727 accounts mentioned previously, which supports our assumption that this community might be composed of compromised accounts that alternate legitimate and malicious activity. The smaller false positive communities, on the other hand, show diurnal patterns similar to the ones observed for legitimate accounts, which support the conclusions that these communities are false positives.

IP addresses and account usage. An alternative representation of account activity over time is to plot the usage graphs for IP addresses and accounts as detailed

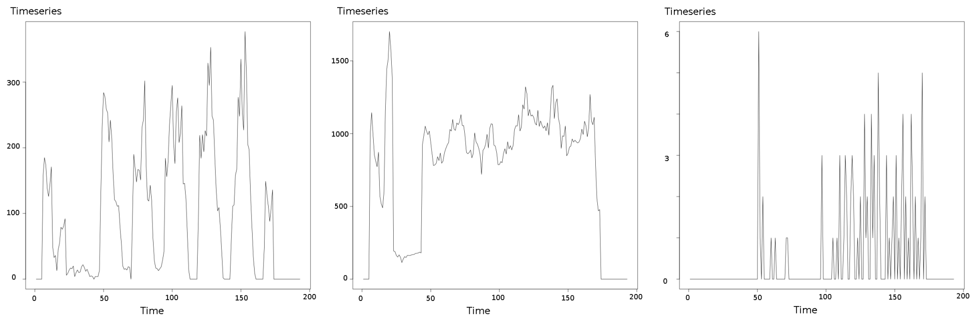

Figure 5: Time series plotting login event over time: legitimate accounts behind a NAT (left plot) and malicious communities (center and right plots).

in Section 3.4. For malicious communities, the usage graphs will exhibit suspicious patterns, indicating synchronization across accounts and IP addresses. For reference, Figure 6 presents the usage graphs for the users behind a NAT. IP address usage is not really relevant for a single IP address, but one can clearly see the daily interruptions over night, as well as the randomness of the events during day time. If we look at malicious communities as plotted in Figure 8, one can see the suspicious vertical patterns appearing. Looking at the IP addresses usage graphs, IP addresses are active in synchronized groups. Previous research already observed that malicious accounts are often used in a synchronized fashion by cybercriminals, and leveraged this property for detection [7]. This gives us additional confidence that the detection performed by EVILCOHORT on the dataset $\mathbf{D}_2$ identifies accounts that are indeed malicious. This anomalous synchronization can be observed for all detected communities of OSN_1, OSN_2, and OSN_3 with the exception of the five potential false positive communities previously mentioned, for which the usage resembles the one of legitimate accounts.

If we look at the large false positive community, however, one can see in Figure 7 that the usage graphs are overall similar to the behavior shown by regular users behind a NAT. However, looking more closely, one can observe multiple darker vertical patterns in the graphs. The mix of legitimate and malicious activities makes us even more confident that such community is indeed composed of compromised accounts accessed by a botnet, and is therefore a true positive detected by EVILCOHORT.

7 Discussion

We showed that EVILCOHORT can be applied to a variety of online services and to any type of activity on these services. This versatility, together with the fact that it complements detections by state-of-the-art systems, makes EVILCOHORT a useful tool in the fight against malicious activity on online services. We hope that, in the future, other researchers will be able to apply the techniques presented in this paper to other online services and types of activity.

As any detection system, EVILCOHORT has some limitations. The main limitation of EVILCOHORT, as we already mentioned, is that it can only detect malicious accounts that are accessed by communities of IP addresses. As we showed in Section 2, however, such accounts are more dangerous than the ones that are accessed by single IP addresses, and existing countermeasures are able to shut down this second type of accounts much quicker.

Another shortcoming is that EVILCOHORT relies on a threshold to limit the number of false positives. An online service that decided to use our approach would have to select a value of s that suits their needs. In this paper we showed that the number of false positives decreases rapidly as we increase s. Applied to our dataset, consisting of millions of events every day, this observation allowed us to reduce false positives to practically zero. Operators can easily tune the value of s by performing sensitivity analysis similar to what we did in Section 5.1.

A last shortcoming is that the accounts used by cybercriminals are not necessarily fake accounts, but could be legitimate accounts that have been compromised. In our current implementation, EVILCOHORT cannot distinguish between the two types of accounts. Dealing with compromised accounts is more difficult, because the online service cannot just suspend them, but has to go through expensive password-reset operations. As we showed in Section 6, it is possible detect whether the detected accounts are fake or compromised by using the postprocessing techniques. A human operator could then decide how to deal with the malicious accounts, depending on their nature.

As with any detection system, a cybercriminal who is aware of EVILCOHORT could attempt to evade it. A straightforward way of doing this would be to have each of the online accounts under his control accessed by a

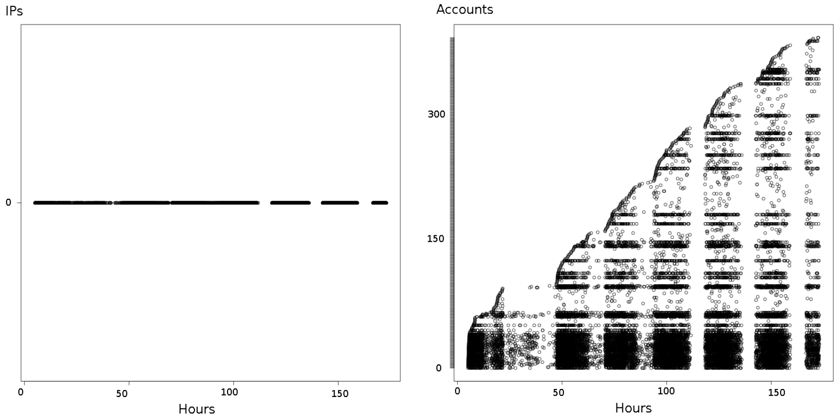

Figure 6: Activity of legitimate users behind a NAT: IP address usage (left) and account usage (right).

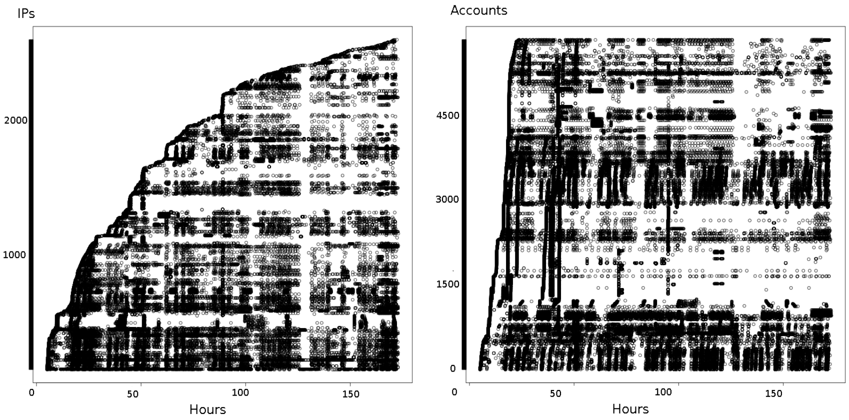

Figure 7: Activity of false positive community: IP address usage (left) and account usage (right). The fact that synchronized activity is interleaved to regular user activity makes us believe that this community is made of compromised accounts.

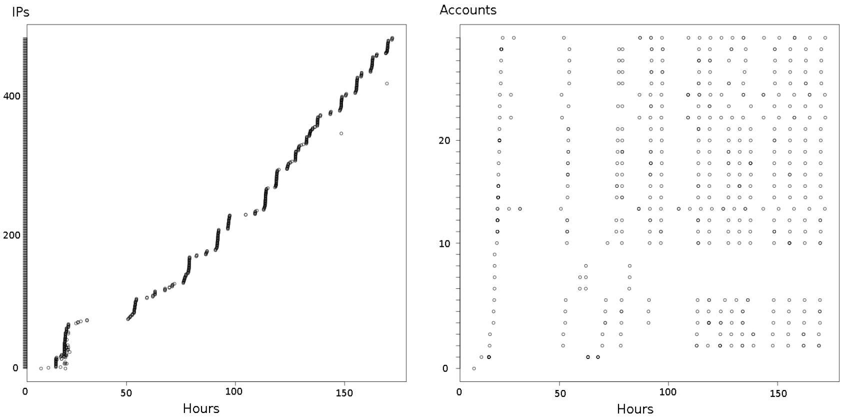

Figure 8: Activity of malicious communities: IP address usage (left) and accounts usage (right).

single IP address. Although this would evade detection by our system, it would make the cybercriminal operation less robust: every time the online service detects and shuts down an account as malicious one of the bots in the botnet becomes useless, while conversely every time a bot is taken offline there is an account on the online service that is not used any more. For this reason EVILCOHORT can be evaded, but by doing so a cybercriminal makes his operation less efficient and profitable.

Since EVILCOHORT can work on any type of activity events, including login events, it has the potential of detecting an account as malicious and blocking it before it performs any malicious activity. In our current implementation, the system works in batches, on time intervals of one day. In the future, we plan to extend it to handle a stream of data, and operate in real time. This way, the system could continuously build communities, and flag an account as malicious as soon as it joins a community.

A source of false alarms for EVILCOHORT could be users who access a service via an IP anonymization service such as Tor. In this case, the set of exit nodes would appear as a community. To mitigate these false positives, we could use the post-processing techniques that we described in Section 3.4. Also, Tor-related communities can be easily identified by comparing the set of IP addresses to the known list of Tor exit nodes. As future work, we plan to explore different tradeoffs between the number of IP addresses accessing a community and the characteristics of such communities.

8 Related Work

A wealth of research has been conducted on detecting malicious activity on online services. Previous work falls into three categories: *content analysis*, *detection of malicious hosts*, and *detection of malicious accounts*.

Content analysis. A corpus of work focuses on detecting malicious content that is shared on online services. Multiple papers dealt with detecting email content that is typical of spam by using machine learning [10, 25]. Other works check whether the URLs posted on an online service are malicious [29,34]. Pitsillidis et al. developed a system that extracts templates from spam [22].

Content analysis systems are effective in detecting and blocking malicious content posted on online services. However, they suffer from two major limitations. The first limitation is that such techniques are typically resource-intensive, and this limits their applicability on busy online services [28]. The second limitation is that such systems can make a detection only when the malicious party tries to posts their content. On the other hand, EVILCOHORT can detect an account (or an IP address) as malicious even if the account does not post any content on the online service.

Detection of malicious hosts (bots). Online services can check in real time if an IP address is known to be a bot by querying DNS blacklists [1]. DNS blacklists are heavily used in anti-spam systems for emails because, unlike content analysis systems, they are lightweight. However, previous research showed that DNS blacklists have a very high number of false negatives [23]. To improve the coverage offered by DNS blacklists, several methods have been proposed. Sinha et al. [24] propose to determine the reputation of an IP address on a global scale, instead of doing it on a local (provider-scale) one. Hao et al. presented SNARE, a system that establishes the reputation of an IP address based on a number of behavioral features (such as the geographical distance between the sender and the recipient) [16].

Detection of malicious accounts. To perform malicious activity on online services, miscreants have to get access to accounts on such services. To this end, they can pay workers to create account for them [33], purchase mass-created fake accounts [30], or buy credentials to compromised accounts on such services [26]. Given the magnitude of the problem, numerous approaches have been proposed to detect accounts that perform malicious activities on online services.

A number of systems analyze the characteristics of accounts on online services, looking for indicators that are typical of mass-created fake accounts (such as the number of people that an account follows) [3, 14, 19, 27, 36, 38]. Yu et al. [37] proposed a system to detect fake social network accounts; the system looks at the network structure, and flags accounts that are not well-connected with their peers in the network as possibly malicious. Benevenuto et al. presented a system to detect accounts that leverage the Youtube service to spread malicious content [4]. Gao et al. [13] developed a system that clusters messages posted on social networks, looking for large-scale spam campaigns. Other approaches look at how messages propagate on social networks, looking for messages that spread anomalously, such as worms [8, 35]. Wang et al. [32] proposed a technique to detect malicious accounts on social networks, based on the sequence of clicks that the people (or the programs) controlling such accounts perform. Jacob et al. [17] presented PUBCRAWL, a system that detects accounts that are used to crawl online services. Egele et al. [11] developed COMPA, a system that learns they typical behavior of social network accounts, and considers any change in behavior as the sign of a possible compromise. Cao et al. presented SynchroTrap [7], a system that detects malicious activity by grouping together social network accounts that performed similar actions during the same period of time. EVILCOHORT improves over SynchroTrap, because accounts do not have to act synchronously to be detected.

Although systems that detect malicious accounts are useful to detect and block malicious activity on online services, they typically rely on a single threat model, and can only detect malicious accounts that operate following that threat model. Conversely, EVILCOHORT leverages the observation that cybercriminals use a set of (compromised) IP addresses and a set of online accounts. For this reason, EVILCOHORT can detect online accounts controlled by cybercriminals, regardless of the purpose for which these accounts are used.

Studying communities of interest. A large corpus of research has been conducted over the years to study communities of interest in networks [2, 20]. Such communities are collections of hosts that share the same goal. Studying communities of interest is useful to model the typical behavior of related hosts, and detect anomalies in their behavior that can be indicative of malicious activity. The communities that we study in this paper are different in nature, because they are composed of online accounts instead of hosts and are consistently used for malicious purposes by miscreants.

The closest work to EVILCOHORT is BOTGRAPH [39]. Similar to EVILCOHORT, this system looks at accounts that are accessed by a common set of IP addresses. However, BOTGRAPH relies on heuristics that are dependent on the email-sending habits of accounts to perform detection, and therefore limit its applicability to the spam-fighting domain. Conversely, EVILCOHORT is principled, and can be applied on any online service without pre-existing domain knowledge. In addition, the fact that EVILCOHORT can be applied on activity other than email-sending events (for example login events) allows us to detect malicious activity other than sending malicious content (e.g., online accounts used as a C&C channel). Another difference is that BOTGRAPH calculates its threshold over a 30-day period, and therefore is not suited to perform detection on freshly-created accounts. In this paper, on the other hand, we showed that EVILCOHORT can work in one-day batches, and detect as malicious accounts that were created during that same day.

9 Conclusions

We presented EVILCOHORT, a system that detects malicious accounts on online services by identifying communities of accounts that are accessed by a common set of computers. Our results show that the vast majority of the accounts that form such communities are used for malicious purposes. In the rare cases in which legitimate communities of accounts form, we show that such communities present characteristics that are very different than the ones of malicious communities. These differences can be used to perform more accurate detection. We ran EVILCOHORT on two real-world datasets, and detected more than one million malicious accounts.

Acknowledgments

This work was supported by the Office of Naval Research (ONR) under grant N00014-12-1-0165, the Army Research Office (ARO) under grant W911NF-09-1-0553, the Department of Homeland Security (DHS) under grant 2009-ST-061-CI0001, the National Science Foundation (NSF) under grant CNS-1408632, and SBA Research.

References

[1] Spamhaus dbl. http://www.spamhaus.org.

[2] AIELLO, W., KALMANEK, C., MCDANIEL, P., SEN, S., SPATSCHECK, O., AND VAN DER MERWE, J. Analysis of communities of interest in data networks. In *Passive and Active Network Measurement*. 2005.

[3] BENEVENUTO, F., MAGNO, G., RODRIGUES, T., AND ALMEIDA, V. Detecting Spammers on Twitter. In *Conference on Email and Anti-Spam (CEAS)* (2010).

[4] BENEVENUTO, F., RODRIGUES, T., ALMEIDA, V., ALMEIDA, J., AND GONÇALVES, M. Detecting spammers and content promoters in online video social networks. In *ACM SIGIR conference on Research and development in information retrieval* (2009).

[5] BILGE, L., STRUFE, T., BALZAROTTI, D., AND KIRDA, E. All Your Contacts Are Belong to Us: Automated Identity Theft Attacks on Social Networks. In *World Wide Web Conference (WWW)* (2009).

[6] BLONDEL, V. D., GUILLAUME, J.-L., LAMBIOTTE, R., AND LEFEBVRE, E. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* (2008).

[7] CAO, Q., YANG, X., YU, J., AND PALOW, C. Uncovering Large Groups of Active Malicious Accounts in Online Social Networks. In *ACM Conference on Computer and Communications Security (CCS)* (2014).

[8] CAO, Y., YEGNESWARAN, V., PORRAS, P., AND CHEN, Y. PathCutter: Severing the Self-Propagation Path of XSS Javascript Worms in Social Web Networks. In *Symposium on Network and Distributed System Security (NDSS)* (2012).

[9] COOKE, E., JAHANIAN, F., AND MCPHERSON, D. The zombie roundup: Understanding, detecting, and disrupting botnets. In *Proceedings of the USENIX SRUTI Workshop* (2005).

[10] DRUCKER, H., WU, D., AND VAPNIK, V. N. Support vector machines for spam categorization. In *IEEE transactions on neural networks* (1999).

[11] EGELE, M., STRINGHINI, G., KRUEGEL, C., AND VIGNA, G. Compa: Detecting compromised accounts on social networks. In *Symposium on Network and Distributed System Security (NDSS)* (2013).

[12] FARKAS, I. J., DERÉNYI, I., BARABÁSI, A.-L., AND VICSEK, T. Spectra of real-world graphs: Beyond the semicircle law. *Physical Review E* (2001).

[13] GAO, H., CHEN, Y., LEE, K., PALSETIA, D., AND CHOUDHARY, A. Towards Online Spam Filtering in Social Networks. In *Symposium on Network and Distributed System Security (NDSS)* (2012).

[14] GAO, H., HU, J., WILSON, C., LI, Z., CHEN, Y., AND ZHAO, B. Detecting and Characterizing Social Spam Campaigns. In *Internet Measurement Conference (IMC)* (2010).

[15] GRIER, C., THOMAS, K., PAXSON, V., AND ZHANG, M. @spam: the underground on 140 characters or less. In *ACM Conference on Computer and Communications Security (CCS)* (2010).

[16] HAO, S., SYED, N. A., FEAMSTER, N., GRAY, A. G., AND KRASSER, S. Detecting Spammers with SNARE: Spatio-temporal Network-level Automatic Reputation Engine. In *USENIX Security Symposium* (2009).

[17] JACOB, G., KIRDA, E., KRUEGEL, C., AND VIGNA, G. Pubcrawl: protecting users and businesses from crawlers. In *USENIX Security Symposium* (2012).

[18] JAGATIC, T., JOHNSON, N., JAKOBSSON, M., AND JAGATIF, T. Social Phishing. *Comm. ACM 50*, 10 (2007), 94–100.

[19] LEE, K., CAVERLEE, J., AND WEBB, S. Uncovering social spammers: social honeypots + machine learning. In *International ACM SIGIR Conference on Research and Development in Information Retrieval* (2010).

[20] MCDANIEL, P. D., SEN, S., SPATSCHECK, O., VAN DER MERWE, J. E., AIELLO, W., AND KALMANEK, C. R. Enterprise Security: A Community of Interest Based Approach. In *Symposium on Network and Distributed System Security (NDSS)* (2006).

[21] NIU, Y., WANG, Y., CHEN, H., MA, M., AND HSU, F. A Quantitative Study of Forum Spamming Using Context-based Analysys. In *Symposium on Network and Distributed System Security (NDSS)* (2007).

[22] PITSILLIDIS, A., LEVCHENKO, K., KREIBICH, C., KANICH, C., VOELKER, G. M., PAXSON, V., WEAVER, N., AND SAVAGE, S. botnet Judo: Fighting Spam with Itself. In *Symposium on Network and Distributed System Security (NDSS)* (2010).

[23] RAMACHANDRAN, A., DAGON, D., AND FEAMSTER, N. Can DNS-based blacklists keep up with bots? In *Conference on Email and Anti-Spam (CEAS)* (2006).

[24] S. SINHA, M. B., AND JAHANIAN, F. Improving Spam Blacklisting Through Dynamic Thresholding and Speculative Aggregation. In *Symposium on Network and Distributed System Security (NDSS)* (2010).

[25] SAHAMI, M., DUMAIS, S., HECKERMANN, D., AND HORVITZ, E. A Bayesian approach to filtering junk e-mail. *Learning for Text Categorization* (1998).

[26] STONE-GROSS, B., COVA, M., CAVALLARO, L., GILBERT, B., SZYDLOWSKI, M., KEMMERER, R., KRUEGEL, C., AND VIGNA, G. Your Botnet is My Botnet: Analysis of a Botnet Takeover. In *ACM Conference on Computer and Communications Security (CCS)* (2009).

[27] STRINGHINI, G., KRUEGEL, C., AND VIGNA, G. Detecting Spammers on Social Networks. In *Annual Computer Security Applications Conference (ACSAC)* (2010).

[28] TAYLOR, B. Sender reputation in a large webmail service. In *Conference on Email and Anti-Spam (CEAS)* (2006).

[29] THOMAS, K., GRIER, C., MA, J., PAXSON, V., AND SONG, D. Design and Evaluation of a Real-Time URL Spam Filtering Service. In *IEEE Symposium on Security and Privacy* (2011).

[30] THOMAS, K. AND MCCOY, D. AND GRIER, C. AND KOLCZ, A. AND PAXSON, V. Trafficking Fraudulent Accounts: The Role of the Underground Market in Twitter Spam and Abuse. In *USENIX Security Symposium* (2013).

[31] THOMASON, A. Blog Spam: A Review. In *Conference on Email and Anti-Spam (CEAS)* (2007).

[32] WANG, G., KONOLIGE, T., WILSON, C., WANG, X., ZHENG, H., AND ZHAO, B. Y. You are how you click: Clickstream analysis for sybil detection. *USENIX Security Symposium* (2013).

[33] WANG, G., WILSON, C., ZHAO, X., ZHU, Y., MOHANLAL, M., ZHENG, H., AND ZHAO, B. Y. Serf and turf: crowdturfing for fun and profit. In *World Wide Web Conference (WWW)* (2012).

[34] XIE, Y., YU, F., ACHAN, K., PANIGRAHY, R., HULTEN, G., AND OSIPKOV, I. Spamming Botnets: Signatures and Characteristics. *SIGCOMM Comput. Commun. Rev.* (2008).

[35] XU, W., ZHANG, F., AND ZHU, S. Toward worm detection in online social networks. In *Annual Computer Security Applications Conference (ACSAC)* (2010).

[36] YANG, C., HARKREADER, R., AND GU, G. Die Free or Live Hard? Empirical Evaluation and New Design for Fighting Evolving Twitter Spammers. In *Symposium on Recent Advances in Intrusion Detection (RAID)* (2011).

[37] YU, H., KAMINSKY, M., GIBBONS, P. B., AND FLAXMAN, A. Sybilguard: defending against sybil attacks via social networks. *ACM SIGCOMM Computer Communication Review* (2006).

[38] ZHANG, C. M., AND PAXSON, V. Detecting and Analyzing Automated Activity on Twitter. In *Passive and Active Measurement Conference* (2011).

[39] ZHAO, Y., XIE, Y., YU, F., KE, Q., YU, Y., CHEN, Y., AND GILLUM, E. Botgraph: Large scale spamming botnet detection. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (2009).

[40] ZHONG, Z., HUANG, K., AND LI, K. Throttling Outgoing SPAM for Webmail Services. In *Conference on Email and Anti-Spam (CEAS)* (2005).