COMPA: Detecting Compromised Accounts on Social Networks

Manuel Egele*†, Gianluca Stringhini*, Christopher Kruegel*, and Giovanni Vigna*

* University of California, Santa Barbara, Santa Barbara, CA
{maeg,gianluca,chris,vigna}@cs.ucsb.edu

† Carnegie Mellon University, Pittsburgh, PA
megele@cmu.edu

Abstract

As social networking sites have risen in popularity, cyber-criminals started to exploit these sites to spread malware and to carry out scams. Previous work has extensively studied the use of fake (Sybil) accounts that attackers set up to distribute spam messages (mostly messages that contain links to scam pages or drive-by download sites). Fake accounts typically exhibit highly anomalous behavior, and hence, are relatively easy to detect. As a response, attackers have started to compromise and abuse legitimate accounts. Compromising legitimate accounts is very effective, as attackers can leverage the trust relationships that the account owners have established in the past. Moreover, compromised accounts are more difficult to clean up because a social network provider cannot simply delete the corresponding profiles.

In this paper, we present a novel approach to detect compromised user accounts in social networks, and we apply it to two popular social networking sites, Twitter and Facebook. Our approach uses a composition of statistical modeling and anomaly detection to identify accounts that experience a sudden change in behavior. Since behavior changes can also be due to benign reasons (e.g., a user could switch her preferred client application or post updates at an unusual time), it is necessary to derive a way to distinguish between malicious and legitimate changes. To this end, we look for groups of accounts that all experience similar changes within a short period of time, assuming that these changes are the result of a malicious campaign that is unfolding. We developed a tool, called COMPA*, that implements our approach, and we ran it on a large-scale dataset of more than 1.4 billion publicly-available Twitter messages, as well as on a dataset of 106 million Facebook messages.* COMPA *was able to identify compromised accounts on both social networks with high precision.*

1 Introduction

Online social networks, such as Facebook and Twitter, have become increasingly popular over the last few years. People use social networks to stay in touch with family, chat with friends, and share news. The users of a social network build, over time, connections with their friends, colleagues, and, in general, people they consider interesting or trustworthy. These connections form a social graph that controls how information spreads in the social network. Typically, users receive messages published by the users they are connected to, in the form of *wall posts*, *tweets*, or *status updates*.

The large user base of these social networks has attracted the attention of cyber-criminals. According to a study from 2008, 83% of social network users received at least one unwanted message on such networks that year [1]. Also, large-scale malware campaigns have been carried out over social networks [2], and previous work has shown that spam, phishing, and malware are real threats on social networking sites [3, 4].

To address the growing problem of malicious activity on social networks, researchers have started to propose different detection and mitigation approaches. Initial work [5, 6, 7] has focused on the detection of *fake accounts* (i.e., automatically created accounts with the sole purpose of spreading malicious content). Unfortunately, systems that detect fake accounts do not discriminate between Sybil and *compromised* accounts. A compromised account is an existing, legitimate account that has been taken over by an attacker. Accounts can be compromised in a number of ways, for example, by exploiting a cross-site scripting vulnerability or by using a phishing scam to steal the user's login credentials. Also, bots have been increasingly used to harvest login information for social networking sites on infected hosts [8].

While dedicated, malicious accounts are easier to cre-

ate, compromised accounts are more valuable to cybercriminals. The reason is that these accounts allow attackers to leverage the associated account history and network of trust to spread malicious content more effectively [9]. As a result, attackers increasingly abuse legitimate accounts to distribute their malicious messages [3, 4]. To identify campaigns that involve both compromised and fake accounts, the focus of the analysis has shifted to the messages themselves. In particular, researchers have proposed techniques that search a social network for the presence of similar messages [10, 3]. The intuition is that attackers send many similar messages as part of campaigns. Similarity is typically defined in terms of overlap in message content or shared URLs.

Of course, it is not sufficient to simply group similar messages to detect malicious campaigns. The reason is that many such clusters (groups) will contain benign messages, which range from simple "happy birthday" wishes to template-based notifications sent by popular applications such as Foursquare [11]. To distinguish benign from malicious clusters, some systems utilize features that are solely based on the URLs in messages [3, 12, 13]. Clearly, these techniques suffer from limited scope, because they cannot find malicious messages that do not contain URLs (we found instances of such messages during our experimental evaluation). Other systems [10] consider additional features such as the size of clusters or the average number of connections for each user. While this broadens their coverage to include campaigns that do not rely on URLs, the reported accuracy of less than 80% is rather sobering. Moreover, and equally important, previous systems can only determine whether a cluster of messages is malicious. That is, they *cannot* distinguish between messages sent by compromised accounts and those sent by fake accounts. This information is crucial for social network operators to initiate appropriate mitigation procedures. Specifically, fake accounts can be safely deleted without affecting legitimate users. To address compromised accounts, however, the social network provider has to notify the victims, reset their passwords, and engage the users in a password recovery process.

In this paper, we present a novel approach to detect compromised accounts on social networks. Our approach offers a combination of three salient features. First, it does not depend on the presence of URLs in messages. As a result, we can detect a broad range of malicious messages, including scam messages that contain telephone numbers and instant messaging contacts. Second, our system is accurate and detects compromised accounts with very low false positives. Third, we focus on finding compromised accounts and leave the detection of fake accounts to previous work [5, 6, 7] or to the social network providers themselves. By identifying compromised accounts, social network providers can focus their mitigation efforts on real users.

The core idea underlying our approach is that it is possible to model the regular activities of individual users. If, at any point, a user's account gets compromised, it is likely that there will be a noticeable change in the account's behavior (and our experiments confirm this assumption). To capture the past behavior of a user, we introduce a collection of statistical models, which we call a *behavioral profile*. Each of our models corresponds to a characteristic feature of a message (e.g., the time of the day when it was sent or the language in which it was written in). These models capture generic user activity on social networks and are not tied to a particular platform (as our experiments on Twitter and Facebook demonstrate). Behavioral profiles make it possible to assess future messages. A message that appears to be very different from a user's typical behavior might indicate a compromise.

Of course, a single message that violates the profile of a user does not necessarily indicate that this account is compromised. The message might be an outlier or merely reflect a normal change in behavior. For this reason, like in previous work, our approach looks for other, similar messages that have recently been posted on the social network and that also violate the behavioral profiles of their respective users. This means that we cannot detect cases in which an attacker posts a single, malicious message through one compromised account. We feel that this is reasonable as attackers typically aim to distribute their malicious messages to a broader victim population. Moreover, our experiments demonstrate that we can detect compromised accounts even in case of small campaigns (in our experiments on Twitter, for example, we require as little as ten similar messages per hour).

In a nutshell, our approach **(i)** checks for a set of similar messages, and **(ii)** requires that a significant subset of these messages violate the behavioral profiles of their senders. These two steps can be executed in any order: We can check for messages that violate their respective behavioral profiles first and then group those messages into clusters of similar ones. This would allow us to implement similarity metrics that are more sophisticated than those presented in previous work (i.e., simple checks for similar content or URLs). Alternatively, we can first group similar messages and then check whether a substantial fraction of these messages violates the corresponding behavioral profiles. Using this order is more efficient as the system has to inspect a smaller number of messages.

We implemented our approach in a system called COMPA. Our system can be used by social network operators to identify compromised accounts and take appropriate countermeasures, such as deleting the offending messages or resetting the passwords of the victims' accounts. Since COMPA relies on behavioral patterns of users and not, like related work, on suspicious message content (URLs [13] or

typical features of Sybil profiles [9]) it is able to detect types of malicious messages that are missed by recently-proposed techniques. In particular, our approach identified scam campaigns that lure their victims into calling a phone number, and hence, the corresponding messages do not contain links (URLs).

We applied COMPA to two large-scale datasets from Twitter and Facebook. The Twitter dataset consists of messages we collected from May 13, 2011 to August 12, 2011, while the Facebook dataset contains messages ranging from September 2007 to July 2009. Our results show that COMPA is effective in detecting compromised accounts with very few false positives. In particular, we detected 383,613 compromised accounts on Twitter, by analyzing three months of data consisting of over 1.4 billion tweets. Furthermore, COMPA detected 11,087 compromised accounts on Facebook, by analyzing 106 million messages posted by users in several large geographic networks.

This paper makes the following contributions:

- We are the first to introduce an approach that focuses on detecting compromised accounts on social networks. This provides crucial input to social network providers to initiate proper mitigation efforts.
- We propose a novel set of features to characterize regular user activity based on the stream of messages that each user posts. We use these features to create models that identify messages that appear anomalous with respect to a user's account (message) history.
- We demonstrate that our approach is able to effectively detect compromised accounts with very low false positives. To this end, we applied our approach to two large-scale datasets obtained from two large social networking sites (Twitter and Facebook).

2 Behavioral Profiles

A behavioral profile leverages historical information about the activities of a social network user to capture this user's normal (expected) behavior. To build behavioral profiles, our system focuses on the stream of messages that a user has posted on the social network. Of course, other features such as profile pictures or friend activity could be useful as well. Unfortunately, social networks typically do not offer a way to retrieve historical data about changes in these features, and therefore, we were unable to use them.

A behavioral profile for a user U is built in the following way: Initially, our system obtains the stream of messages of U from the social networking site. The message stream is a list of all messages that the user has posted on the social network, in chronological order. For different social networks, the message streams are collected in slightly different ways. For example, on Twitter, the message stream corresponds to a user's public timeline. For Facebook, the message stream contains the posts a user wrote on her own wall, but it also includes the messages that this user has posted on her friends' walls.

To be able to build a comprehensive profile, the stream needs to contain a minimum amount of messages. Intuitively, a good behavioral profile has to capture the breadth and variety of ways in which a person uses her social network account (e.g., different client applications or languages). Otherwise, an incomplete profile might incorrectly classify legitimate user activity as anomalous. Therefore, we do not create behavioral profiles for accounts whose stream consists of less than a minimum number S of messages. In our experiments, we empirically determined that a stream consisting of less than $S = 10$ messages does usually not contain enough variety to build a representative behavioral profile for the corresponding account. Furthermore, profiles that contain less then S messages pose a limited threat to the social network or its users. This is because such accounts are either new or very inactive and thus, their contribution to large scale campaigns is limited. A detailed discussion of this threshold is provided in Section 6.

Once our system has obtained the message stream for a user, we use this information to build the corresponding behavioral profile. More precisely, the system extracts a set of feature values from each message, and then, for each feature, trains a statistical model. Each of these models captures a characteristic feature of a message, such as the time the message was sent, or the application that was used to generate it. The features used by these models, as well as the models themselves, are described later in this section.

Given the behavioral profile for a user, we can assess to what extent a new message corresponds to the expected behavior. To this end, we compute the anomaly score for a message with regard to the user's established profile. The anomaly score is computed by extracting the feature values for the new message, and then comparing these feature values to the corresponding feature models. Each model produces a score (real value) in the interval $[0, 1]$, where 0 denotes perfectly normal (for the feature under consideration) and 1 indicates that the feature is highly anomalous. The anomaly score for a message is then calculated by composing the results for all individual models.

2.1 Modeling Message Characteristics

Our approach models the following seven features when building a behavioral profile.

Time (hour of day). This model captures the hour(s) of the day during which an account is typically active. Many users have certain periods during the course of a day where they are more likely to post (e.g., lunch breaks) and others that are typically quiet (e.g., regular sleeping hours). If a

user's stream indicates regularities in social network usage, messages that appear during hours that are associated with quiet periods are considered anomalous.

Message Source. The source of a message is the name of the application that was used to submit it. Most social networking sites offer traditional web and mobile web access to their users, along with applications for mobile platforms such as iOS and Android. Many social network ecosystems provide access to a multitude of applications created by independent, third-party developers.

Of course, by default, a third-party application cannot post messages to a user's account. However, if a user chooses to, she can grant this privilege to an application. The state-of-the-art method of governing the access of third-party applications is OAUTH [14]. OAUTH is implemented by Facebook and Twitter, as well as numerous other, high-profile web sites, and enables a user to grant access to her profile without revealing her credentials.

By requiring all third-party applications to implement OAUTH, the social network operators can easily shut down individual applications, should that become necessary. In fact, our evaluation shows that third-party applications are frequently used to send malicious messages.

This model determines whether a user has previously posted with a particular application or whether this is the first time. Whenever a user posts a message from a new application, this is a change that could indicate that an attacker has succeeded to lure a victim into granting access to a malicious application.

Message Text (Language). A user is free to author her messages in any language. However, we would expect that each user only writes messages in a few languages (typically, one or two). Thus, especially for profiles where this feature is relatively stable, a change in the language is an indication of a suspicious change in user activity.

To determine the language that a message was written in, we leverage the `libtextcat` library. This library performs n-gram-based text categorization, as proposed by Cavnar and Trenkle [15]. Of course, for very short messages, it is often difficult to determine the language. This is particularly problematic for Twitter messages, which are limited to at most 140 characters and frequently contain abbreviated words or uncommon spelling.

Message Topic. Users post many messages that contain chatter or mundane information. But we would also expect that many users have a set of topics that they frequently talk about, such as favorite sports teams, music bands, or TV shows. When users typically focus on a few topics in their messages and then suddenly post about some different and unrelated subject, this new message should be rated as anomalous.

In general, inferring message topics from short snippets of text without context is difficult. However, some social networking platforms allow users to label messages to explicitly specify the topics their messages are about. When such labels or tags are available, they provide a valuable source of information. A well-known example of a message-tagging mechanism are Twitter's *hashtags*. By prefixing the topic keyword with a hash character a user would use #Olympics to associate her tweet with the Olympic Games.

More sophisticated (natural language processing) techniques to extract message topics are possible. However, such techniques are outside the scope of this paper.

Links in Messages. Often, messages posted on social networking sites contain links to additional resources, such as blogs, pictures, videos, or news articles. Links in messages of social networks are so common that some previous work has strongly focused on the analysis of URLs, often as the sole factor, to determine whether a message is malicious or not. We also make use of links as part of the behavioral profile of a user, but only as a single feature. Moreover, recall that our features are primarily concerned with capturing the normal activity of users. That is, we do not attempt to detect whether a URL is malicious in itself but rather whether a link is different than what we would expect for a certain user.

To model the use of links in messages, we only make use of the domain name in the URL of links. The reason is that a user might regularly refer to content on the same domain. For example, many users tend to read specific news sites and blogs, and frequently link to interesting articles there. Similarly, users have preferences for a certain URL shortening service. Of course, the full link differs among these messages (as the URL path and URL parameters address different, individual pages). The domain part, however, remains constant. Malicious links, on the other hand, point to sites that have no legitimate use. Thus, messages that link to domains that have not been observed in the past indicate a change. The model also considers the general frequency of messages with links, and the consistency with which a user links to particular sites.

Direct User Interaction. Social networks offer mechanisms to directly interact with an individual user. The most common way of doing this is by sending a direct message that is addressed to the recipient. Different social networks have different mechanisms for doing that. For example, on Facebook, one posts on another user's wall; on Twitter, it is possible to directly "mention" other users by putting the @ character before the recipient's user name. Over time, a user builds a personal interaction history with other users on the social network. This feature aims to capture the interaction history for a user. In fact, it keeps track of the users an account ever interacted with. Direct messages are sent to catch the attention of their recipients, and thus are frequently used by spammers.

Proximity. In many cases, social network users befriend other users that are close to them. For example, a typical Facebook user will have many friends that live in the same city, went to the same school, or work for the same company. If this user suddenly started interacting with people who live on another continent, this could be suspicious. Some social networking sites (such as Facebook) express this proximity notion by grouping their users into networks. The proximity model looks at the messages sent by a user. If a user sends a message to somebody in the same network, this message is considered as local. Otherwise, it is considered as not local. This feature captures the fraction of local vs. non-local messages.

If COMPA is implemented directly by a social network provider, the geo-locations of the users' IP addresses can be used to significantly improve the proximity feature. Unfortunately, this information is not available to us.

3 Detecting Anomalous Messages

3.1 Training and Evaluation of the Models

In this section, we first discuss how we train models for each of the previously-introduced features. We then describe how we apply a model to a new message to compute an anomaly score. Finally, we discuss how the scores of the different models are combined to reach a final anomaly score that reflects how the new message is different from the historic messages used when building the model.

Training. The input for the training step of a model is the series of messages (the message stream) that were extracted from a user account. For each message, we extract the relevant features such as the source application and the domains of all links.

Each feature model is represented as a set **M**. Each element of **M** is a tuple $< fv, c >$. fv is the value of a feature (e.g., `English` for the language model, or `example.com` for the link model). c denotes the number of messages in which the specific feature value fv was present. In addition, each model stores the total number N of messages that were used for training.

Our models fall into two categories:

- *Mandatory* models are those where there is one feature value for each message, and this feature value is always present. Mandatory models are *time of the day*, *source*, *proximity*, and *language*.

- *Optional* models are those for which not every message has to have a value. Also, unlike for mandatory models, it is possible that there are multiple feature values for a single message. Optional models are *links*, *direct interaction*, and *topic*. For example, it is possible that a message contains zero, one, or multiple links. For each optional model, we reserve a specific element with fv = `null`, and associate with this feature value the number of messages for which no feature value is present (e.g., the number of messages that contain no links).

The training phase for the *time of the day* model works slightly differently. Based on the previous description, our system would first extract the hour of the day for each message. Then, it would store, for each hour fv, the number of messages that were posted during this hour. This approach has the problem that hour slots, unlike the progression of time, are discrete. Therefore, messages that are sent close to a user's "normal" hours could be incorrectly considered as anomalous.

To avoid this problem, we perform an adjustment step after the *time of the day* model was trained (as described above). In particular, for each hour i, we consider the values for the two adjacent hours as well. That is, for each element $< i, c_i >$ of **M**, a new count $c\prime_i$ is calculated as the average between the number of messages observed during the ith hour (c_i), the number of messages sent during the previous hour (c_{i-1}), and the ones observed during the following hour (c_{i+1}). After we computed all $c\prime_i$, we replace the corresponding, original values in **M**.

As we mentioned previously, we cannot reliably build a behavioral profile if the message stream of a user is too short. Therefore, the training phase is aborted for streams shorter than $S = 10$, and any message sent by those users is not evaluated.

Evaluating a new message. When calculating the anomaly score for a new message, we want to evaluate whether this message violates the behavioral profile of a user for a given model. In general, a message is considered more anomalous if the value for a particular feature did not appear at all in the stream of a user, or it appeared only a small number of times. For *mandatory features*, the anomaly score of a message is calculated as follows:

1. The feature fv for the analyzed model is first extracted from the message. If **M** contains a tuple with fv as a first element, then the tuple $< fv, c >$ is extracted from **M**. If there is no tuple in **M** with fv as a first value, the message is considered anomalous. The procedure terminates here and an anomaly score of 1 is returned.

2. As a second step, the approach checks if fv is anomalous at all for the behavioral profile being analyzed. c is compared to $\bar{M}$, which is defined as $\bar{M} = \frac{\sum_{i=1}^{\|\mathbf{M}\|} c_i}{N}$, where c_i is, for each tuple in **M**, the second element of the tuple. If c is greater or equal than $\bar{M}$, the message is considered to comply with the learned behavioral

profile for that model, and an anomaly score of 0 is returned. The rationale behind this is that, in the past, the user has shown a significant number of messages with that particular fv.

3. If c is less than $\bar{M}$, the message is considered somewhat anomalous with respect to that model. Our approach calculates the relative frequency f of fv as $f = \frac{c_{fv}}{N}$. The system returns an anomaly score of 1 - f.

The anomaly score for *optional features* is calculated as:

1. The feature fv for the analyzed model is first extracted from the message. If **M** contains a tuple with fv as a first element, the message is considered to match the behavioral profile, and an anomaly score of 0 is returned.

2. If there is no tuple in **M** with fv as a first element, the message is considered anomalous. The anomaly score in this case is defined as the probability p for the account to have a `null` value for this model. Intuitively, if a user rarely uses a feature on a social network, a message containing an fv that has never been seen before for this feature is highly anomalous. The probability p is calculated as $p = \frac{c_{null}}{N}$. If **M** does not have a tuple with `null` as a first element, c_{null} is considered to be 0. p is then returned as the anomaly score.

As an example, consider the following check against the *language* model: The stream of a particular user is composed of 21 messages. Twelve of them are in English, while nine are in German. The **M** of the user for that particular model looks like this:

(<English,12>,<German,9>).

The next message sent by that user will match one of three cases:

- The new message is in English. Our approach extracts the tuple <English,12> from **M**, and compares $c = 12$ to $\bar{M} = 10.5$. Since c is greater than $\bar{M}$, the message is considered normal, and an anomaly score of 0 is returned.

- The new message is in Russian. Since the user never sent a message in that language before, the message is considered very suspicious, and an anomaly score of 1 is returned.

- The new message is in German. Our approach extracts the tuple <German, 9> from **M**, and compares $c = 9$ to $\bar{M} = 10.5$. Since $c < \bar{M}$, the message is considered slightly suspicious. The relative frequency of German tweets for the user is $f = \frac{c}{N} = 0.42$. Thus, an anomaly score of $1 - f = 0.58$ is returned. This means that the message shows a slight anomaly in the user average behavior. However, as explained in Section 5, on its own this score will not be enough to flag the message as malicious.

Computing the final anomaly score. Once our system has evaluated a message against each individual model, we need to combine the results into an overall anomaly score for this message. This anomaly score is a weighted sum of the values for all models. We use Sequential Minimal Optimization [16] to learn the optimal weights for each model, based on a training set of instances (messages and corresponding user histories) that are labeled as malicious and benign. Of course, different social networks will require different weights for the various features. A message is said to violate an account's behavioral profile if its overall anomaly score exceeds a threshold. In Section 5, we present a more detailed discussion on how the threshold values are determined. Moreover, we discuss the weights (and importance) of the features for the different social networks that we analyzed (i.e., Twitter and Facebook).

3.2 Robustness of the Models

Malicious campaigns that are executed through compromised accounts will, in general, fail to match the expected behavior of a vast majority of their victims. One reason is that it is very difficult for an attacker to make certain features look normal, even if the attacker has detailed knowledge about the history of a victim account. In particular, this is true for the *application source* and the *links* features. Consider a user who always posts using her favorite Twitter client (e.g., from her iPhone). Since the attacker does not control this third-party application, and the social network (Twitter) automatically appends the source information to the message, a malicious message will not match the victim's history. Furthermore, to send messages from an iPhone application, the attacker would have to instrument a physical iPhone device to log into the victims' accounts and post the malicious messages. Clearly, such an attack model does not scale. To satisfy the link model, an attacker would need to host his malicious page on a legitimate, third-party domain (one of the domains that the user has linked to in the past). It is very unlikely that an attacker can compromise arbitrary third-party sites that the different victims have referenced in the past.

Other feature models can be matched more easily, assuming that the attacker has full knowledge about the history of a victim account. In particular, it is possible to post at an expected time, use a language that the victim has used in the past, and craft the message so that both the topic and direct user interactions match the observed history. However, crafting customized messages is very

resource-intensive. The reason is that this would require the attacker to gather the message history for all victim users. Since social network sites typically rate-limit access to user profiles, gathering data for many victims is a non-trivial endeavor (we initially faced similar limitations when performing our experiments; and we had to explicitly ask the social networks to white-list our IP addresses).

The need to customize messages makes it also more difficult to coordinate large-scale attacks. First, it requires delaying messages for certain victims until an appropriate time slot is reached. This could provide more time for the social network to react and take appropriate countermeasures. Second, when messages have different topics, attackers cannot easily perform search engine optimizations or push popular terms, simply because victim users might not have used these terms in the past. Also, the proximity feature can help limiting the spread of a campaign. If a user always messages users that are close to her, the number of possible victims is reduced. Of course, the burden for the attacker to blend his messages into the stream of his victims decreases with the number of victims. That is, targeted attacks against individuals or small groups are more challenging to detect. However, the precision of the behavioral profiles that COMPA generates (see Section 6.5) makes us confident that similar mechanisms can contribute to the problem of identifying such small-scale targeted attacks.

Overall, given the challenges to make certain features appear normal and the practical difficulties to craft customized messages to satisfy the remaining models, our feature set is robust with regard to large-scale attacks that leverage compromised accounts. Our experiments show that COMPA is successful in identifying campaigns that use compromised accounts to distribute malicious messages.

3.3 Novelty of the modelled features

We also compared our features with respect to existing work. However, the purpose of our system is very different from the goals of the ones proposed in previous work. These systems generally aim at detecting accounts that have been specifically created to spread spam or malicious content [5, 6, 7, 17]. Since these accounts are controlled in an automated fashion, previous systems detect accounts that always act in a similar way. Instead, we look for sudden changes in behavior of legitimate but compromised social network accounts. Table 1 lists in detail the features previous systems used to achieve their goals, and compares them to the features used by our system. In particular, we studied the works from Benvenuto et al. [5], Gao et al. [10], Grier et al. [4], Lee et al. [6], Stringhini et al. [7], Yang et al. [17], Cai et al. [18], and Song et al. [19].

As it can be seen, our system does not use any of the *Network Features* or any of the *Friends Features*. Such features aim to detect whether a certain account has been created automatically, therefore, they are not useful for our purpose. The reason is that, since the profiles we want to detect are legitimate ones that got compromised, these features would look normal for such profiles. Also, we do not use any of the *Single Message Features*. These features aim to detect a malicious message when it contains words that are usually associated with malicious content (e.g., *cheap drugs*), or when the URL is listed in a blacklist such as SURBL [20]. Since we did not want to limit our approach to flagging messages that contain known malicious sites or well-known words, we did not include such features in our models. In the future, we could use these features to improve our system.

In COMPA, we focus on *Stream Features*. These features capture the characteristics of a user's message stream, such as the ratio of messages that include links, or the similarity among the messages. Looking at Table 1, it seems that five of our features (except the *Language* and *Proximity* features) have been previously used by at least one other system. However, the way these systems use such features is the opposite of what we do: Previous work wants to detect *similarity*, while we are interested in *anomalies*. For example, the *message timing feature* has been used by Grier et al. [4], by Gao et al. [10], and by COMPA for building the *time of the day* model. However, what previous work is looking for are profiles that show a high grade of automation (by looking for profiles that send messages at the same minute every hour), or for short-lived, bursty spam campaigns. Instead, we want to find profiles that start posting at unusual times.

Only the *user interactions* feature has been used in a similar fashion by another system. Gao et al. [3] use it as indication of possibly compromised accounts. Similarly to our system, they flag any account that ever had a legitimate interaction with another user, and started sending malicious content at a later time. However, they identify "malicious content" based only on URL blacklists and suspicious words in the messages. Thus, they are much more limited in their detection capabilities, and their approach mislabels fake profiles that try to be stealthy by sending legitimate-looking messages.

4 Grouping of Similar Messages

A single message that violates the behavioral profile of a user does not necessarily indicate that this user is compromised and the message is malicious. The message might merely reflect a normal change of behavior. For example, a user might be experimenting with new client software or expanding her topics of interest. Therefore, before we flag an account as compromised, we require that we can find a number of similar messages (within a specific time interval) that also violate the accounts of their respective senders.

This means that we cannot detect cases in which an at-

	[5]	[3]	[4]	[6]	[7]	[17]	[18]	[19]	**COMPA**
Network Features									
Avg # conn. of neighbors						✔			
Avg messages of neighbors						✔			
Friends to Followers (F2F)	✔	✔			✔				
F2F of neighbors						✔			
Mutual links						✔	✔	✔	
User distance								✔	
Single Message Features									
Suspicious content	✔								
URL blacklist			✔						
Friends features									
Friend name entropy					✔				
Number of friends	✔				✔				
Profile age	✔								
Stream Features									
Activity per day	✔								
Applications used						✔			✔
Following Rate						✔			
Language									✔
Message length	✔								
Messages sent					✔				
Message similarity		✔	✔	✔	✔	✔			
Message timing		✔	✔						✔
Proximity									✔
Retweet ratio	✔								
Topics	✔								✔
URL entropy			✔						
URL ratio	✔	✔		✔	✔	✔			
URL repetition				✔					✔
User interaction	✔	✔		✔					✔

Table 1. Comparison of the features used by previous work

tacker posts a single, malicious message through one compromised account. While it is very possible that our models would correctly identify that message as suspicious, alerting on all behavioral profile violations results in too many false positives. Hence, we use message similarity as a second component to distinguish malicious messages from spurious profile violations. This is based on the assumption that attackers aim to spread their malicious messages to a larger victim population. However, it is important to note that this does not limit COMPA to the detection of large-scale campaigns. In our experiments on the Twitter platform, for example, we only require ten similar messages per hour before reporting accounts as compromised.

As mentioned previously, we can either first group similar messages and then check all clustered messages for behavioral profile violations, or we can first analyze all messages on the social network for profile violations and then cluster only those that have resulted in violations. The latter approach offers more flexibility for grouping messages, since we only need to examine the small(er) set of messages that were found to violate their user profiles. This would allow us to check if a group of suspicious messages was sent by users that are all directly connected in the social graph, or whether these messages were sent by people of a certain demographics. Unfortunately, this approach requires to check *all* messages for profile violations. While this is certainly feasible for the social networking provider, our access to these sites is rate-limited in practice. Hence, we need to follow the first approach: More precisely, we first group similar messages. Then, we analyze the messages in clusters for profile violations. To group messages, we use the two simple similarity measures, discussed in the following paragraphs.

Content similarity. Messages that contain similar text can be considered related and grouped together. To this end, our first similarity measure uses n-gram analysis of a message's text to cluster messages with similar contents. We use entire words as the basis for the n-gram analysis. Based on initial tests to evaluate the necessary computational resources and the quality of the results, we decided to use four-grams. That is, two messages are considered similar if they share at least one four-gram of words (i.e., four consecutive, identical words).

URL similarity. This similarity measure considers two messages to be similar if they both contain at least one link to a similar URL. The naïve approach for this similarity measure would be to consider two messages similar if they contain an identical URL. However, especially for spam campaigns, it is common to include identifiers into the query string of a URL (i.e., the part in a URL after the question mark). Therefore, this similarity measure discards the query string and relies on the remaining components of a URL to assess the similarity of messages. Of course, by discarding the query string, the similarity measure might be incorrectly considering messages as similar if the target site makes use of the query string to identify different content. Since `YouTube` and `Facebook` use the query string to address individual content, this similarity measure discards URLs that link to these two sites.

Many users on social networking sites use URL shortening services while adding links to their messages. In principle, different short URLs could point to the same page, therefore, it would make sense to expand such URLs, and perform the grouping based on the expanded URLs. Unfortunately, for performance reasons, we could not expand short URLs in our experiments. On Twitter, we observe several million URLs per day (most of which are shortened). This exceeds by far the limits imposed by any URL shortening service.

We do not claim that our two similarity measures represent the only ways in which messages can be grouped. However, as the evaluation in Section 6 shows, the similarity measures we chose perform very well in practice. Furthermore, our system can be easily extended with additional similarity measures if necessary.

5 Compromised Account Detection

Our approach groups together similar messages that are generated in a certain time interval. We call this the *observation interval*. For each group, our system checks all accounts to determine whether each message violates the corresponding account's behavioral profile. Based on this analysis, our approach has to make a final decision about whether an account is compromised or not.

Suspicious groups. A group of similar messages is called a *suspicious group* if the fraction of messages that violates their respective accounts' behavioral profiles exceeds a threshold th. In our implementation, we decided to use a threshold that is dependent on the size of the group. The rationale behind this is that, for small groups, there might not be enough evidence of a campaign being carried out unless a high number of similar messages violate their underlying behavioral profiles. In other words, small groups of similar messages could appear coincidentally, which might lead to false positives if the threshold for small groups is too low. This is less of a concern for large groups that share a similar message. In fact, even the existence of large groups is already somewhat unusual. This can be taken into consideration by choosing a lower threshold value for larger groups. Accordingly, for large groups, it should be sufficient to raise an alert if a smaller percentage of messages violate their behavioral profiles. Thus, the threshold th is a linear function of the size of the group n defined as $th(n) = \max(0.1, kn + d)$.

Based on small-scale experiments, we empirically determined that the parameters $k = -0.005$ and $d = 0.82$ work well. The max expression assures that at least ten percent of the messages in big groups violate their behavioral profiles. Our experiments show that these threshold values are robust, as small modifications do not influence the quality of the results. Whenever there are more than th messages in a group (where each message violates its profile), COMPA declares all users in the group as compromised.

Bulk applications. Certain popular applications, such as Nike+ or Foursquare, use templates to send similar messages to their users. Unfortunately, this can lead to false positives. We call these applications bulk applications. To identify popular bulk applications that send very similar messages in large amounts, COMPA needs to distinguish regular client applications (which do not automatically post using templates) from bulk applications. To this end, our system analyzes a randomly selected set of S messages for each application, drawn from *all* messages sent by this application. COMPA then calculates the average pairwise Levenshtein ratios for these messages. The Levenshtein ratio is a measure of the similarity between two strings based on the edit distance. The values range between 0 for unrelated strings and 1 for identical strings. We empirically determined that the value 0.35 effectively separates client from bulk applications.

COMPA flags all suspicious groups produced by client applications as compromised. For bulk applications, a further distinction is necessary, since we only want to discard groups that are due to *popular* bulk applications. Popular bulk applications constantly recruit new users. Also, these messages are commonly synthetic, and they often violate the behavioral profiles of new users. For existing users, on the other hand, past messages from such applications contribute to their behavioral profiles, and thus, additional messages do not indicate a change in behavior. If many users made use of the application in the past, and the messages the application sent were in line with these users' behavioral profiles, COMPA considers such an application as popular.

To assess an application's popularity, COMPA calculates the number of distinct accounts in the social network that made use of that application before it has sent the first message that violates a user's behavioral profile. This number is multiplied by an age factor (which is the number of seconds between the first message of the application as observed by COMPA and the first message that violated its user's behavioral profile). The intuition behind this heuristics is the following: An application that has been used by many users for a long time should not raise suspicion when a new user starts using it, even if it posts content that differs from this user's established behavior. Manual analysis indicated that bulk applications that are used to run spam and phishing campaigns over compromised accounts have a very low popularity score. Thus, COMPA considers a bulk application to be popular if its score is above 1 million. We assume that popular bulk applications do not pose a threat to their users. Consequently, COMPA flags a suspicious group as containing compromised accounts only if the group's predominant application is a non-popular bulk application.

6 Evaluation

We implemented our approach in a tool, called COMPA and evaluated it on Twitter and Facebook; we collected tweets in real time from Twitter, while we ran our Facebook experiments on a large dataset crawled in 2009.

We show that our system is capable of building meaningful behavioral profiles for individual accounts on both networks. By comparing new messages against these profiles, it is possible to detect messages that represent a (possibly malicious) change in the behavior of the account. By grouping together accounts that contain similar messages, many of which violate their corresponding accounts' behavioral profiles, COMPA is able to identify groups of compromised accounts that are used to distribute malicious messages on these social networks. We continuously ran COMPA on a stream of 10% of all public Twitter messages on a single

computer (Intel Xeon X3450, 16 GB ram). The main limitation was the number of user timelines we could request from Twitter, due to the enforced rate-limits. Thus, we are confident that COMPA can be scaled up to support online social networks of the size of Twitter with moderate hardware requirements.

6.1 Data Collection

Twitter Dataset

We obtained elevated access to Twitter's streaming and RESTful API services. This allowed us to collect around 10% of all public tweets through the streaming API, resulting in roughly 15 million tweets per day on average. We collected this data continuously starting May 13, 2011 until Aug 12, 2011. In total, we collected over 1.4 billion tweets from Twitter's stream. The stream contains live tweets as they are sent to Twitter. We used an observation interval of one hour. Note that since the stream contains randomly sampled messages, COMPA regenerated the behavioral profiles for all involved users every hour. This was necessary, because it was not guaranteed that we would see the same user multiple times.

To access the historical timeline data for individual accounts, we rely on the RESTful API services Twitter provides. To this end, Twitter whitelisted one of our IP addresses, which allowed us to make up to 20,000 RESTful API calls per hour. A single API call results in at most 200 tweets. Thus, to retrieve complete timelines that exceed 200 tweets, multiple API requests are needed. Furthermore, Twitter only provides access to the most recent 3,200 tweets in any user's timeline. To prevent wasting API calls on long timelines, we retrieved timeline data for either the most recent three days, or the user's 400 most recent tweets, whatever resulted in more tweets.

On average, we received tweets from more than 500,000 distinct users per hour. Unfortunately, because of the API request limit, we were not able to generate profiles for all users that we saw in the data stream. Thus, as discussed in the previous section, we first cluster messages into groups that are similar. Then, starting from the largest cluster, we start to check whether the messages violate the behavioral profiles of their senders. We do this, for increasingly smaller clusters, until our API limit is exhausted. On average, the created groups consisted of 30 messages. This process is then repeated for the next observation period.

Facebook Dataset

Facebook does not provide a convenient way of collecting data. Previous work deployed honey accounts on Facebook, and collected data for the accounts that contacted them [7]. Unfortunately, this approach does not scale, since a very large number of honey profiles would be required to be able to collect a similar number of messages as we did for Twitter. Therefore, we used a dataset that was crawled in 2009. We obtained this dataset from an independent research group that performed the crawling in accordance with the privacy guidelines at their research institution. Unfortunately, Facebook is actively preventing researchers from collecting newer datasets from their platform by various means, including the threat of legal action.

The dataset was crawled from geographic networks on Facebook. Geographic networks were used to group together people that lived in the same area. The default privacy policy for these networks was to allow anybody in the network to see all the posts from all other members. Therefore, it was easy, at the time, to collect millions of messages by creating a small number of profiles and join one of these geographic networks. For privacy reasons, geographic networks have been discontinued in late 2009. The dataset we used contains 106,373,952 wall posts collected from five geographic networks (i.e., London, New York, Los Angeles, Monterey Bay, and Santa Barbara). These wall posts are distributed over almost two years (Sept. 2007 - July 2009).

6.2 Training the Classifier

To determine the weights that we have to assign to each feature, we applied Weka's SMO [21] to a labeled training dataset for both Twitter and Facebook.

While the Facebook dataset contains the network of a user, Twitter does not provide such a convenient proximity feature. Therefore, we omitted this feature from the evaluation on Twitter. For Twitter, the weights for the features are determined from our labeled training dataset consisting of 5,236 (5142 legitimate, 94 malicious) messages with their associated feature values as follows: *Source (3.3)*, *Personal Interaction (1.4)*, *Domain (0.96)*, *Hour of Day (0.88)*, *Language (0.58)*, and *Topic (0.39)*.

To manually determine the ground truth for an account in our training set, we examined the tweets present in that account's timeline. If an account under analysis uses URLs in tweets, we follow these links and inspect the landing pages. Should we find that the URL lead to a phishing page, we classify the account as compromised. As we discuss in Section 6.6, phishing campaigns frequently make use of URLs to guide potential victims to phishing websites that prompt the visitor to disclose her account credentials. Another source of information we used to assess whether an account was compromised are application description pages. Each tweet sent by a third-party application contains a link to a website chosen by the developer. If such a link leads to a malicious page, we also consider the account as compro-

mised [1]. Finally, we exploit the fact that humans can extract the topic of a message from small amounts of information. That is, we would flag an account as compromised if the topic of tweets in the timeline abruptly switches from personal status updates to tweets promoting work from home opportunities and free electronic gadgets (common scams). As we will show later in this section, a significant portion of the tweets that indicate that an account is compromised get removed. This makes it time consuming to manually identify compromised accounts on Twitter. Although the number of malicious samples in our training dataset is limited, the feature values turned out to be stable over different training set sizes.

Figure 1 illustrates how the weights for each feature vary with different sizes of the training set. Each set of five bars corresponds to one feature. Each bar within a set represents the observed weights for this feature (i.e., *average*, *min*, and *max*) that were produced by 25 iterations with a fixed training set size. For each iteration, the contents of the training set were randomly chosen. Overall, this experiment was repeated five times with different training set sizes. It can be seen that when smaller training sets are used, the observed weights vary heavily. This variance becomes small for lager training datasets indicating that the weights are fairly stable.

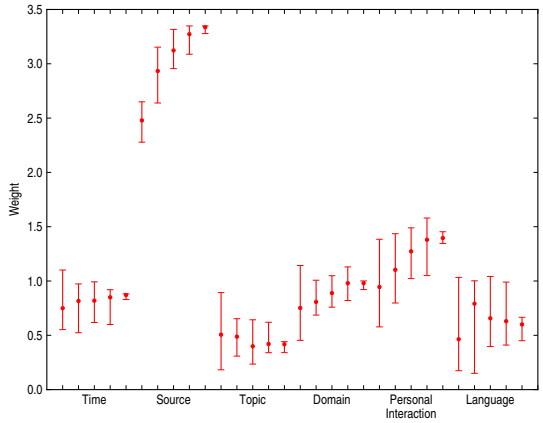

Figure 1. Features evolving with different sizes of training sets. Each experiment was conducted 25 times on random subsets of 25%, 50%, 70%, 90%, and 99% of the 5,236 labeled training instances. The fraction of positive to negative samples remained constant.

On Facebook, based on a labeled training dataset of 279 messages (181 legitimate, 122 malicious), the weights were: *Source (2.2)*, *Domain (1.1)*, *Personal Interaction (0.13)*, *Proximity (0.08)*, and *Hour of Day (0.06)*. Weka determined that the *Language* feature has no effect on the classification. Moreover, as discussed earlier, assessing the message topic of an unstructured message is a complicated natural language processing problem. Therefore, we omitted this feature from the evaluation on the Facebook dataset. Similar to analyzing Twitter messages, we also assessed changes of topic across wall posts in the Facebook dataset to identify compromised accounts for the training data. Additionally, we inspected Facebook application pages and their comment sections where users can leave feedback. As the Facebook dataset was collected in 2009, we would also consider an account as compromised if the application that sent that post was blocked by Facebook in the meantime.

[1] While Twitter has been filtering links to potentially malicious URLs in posted messages for a while, they only started filtering these application pages after *we* informed Twitter that an attacker can choose this page to be a malicious site.

6.3 Detection on Twitter

The overall results for our Twitter evaluation are presented in Table 2. Due to space constraints, we will only discuss the details for the *text similarity measure* here. However, we found considerable overlap in many of the groups produced by both similarity measures. More precisely, for over 8,200 groups, the two similarity measures (content and URL similarity) produced overlaps of at least eight messages. COMPA found, for example, phishing campaigns that use the same URLs and the same text in their malicious messages. Therefore, both similarity measures produced overlapping groups.

The *text similarity* measure created 374,920 groups with messages of similar content. 365,558 groups were reported as legitimate, while 9,362 groups were reported as compromised. These 9,362 groups correspond to 343,229 compromised accounts. Interestingly, only 12,238 of 302,513 applications ever produced tweets that got grouped together. Furthermore, only 257 of these applications contributed to the groups that were identified as compromised.

For each group of similar messages, COMPA assessed whether the predominant application in this group was a regular client or a bulk application. Our system identified 12,347 groups in the bulk category, of which 1,647 were flagged as compromised. Moreover, COMPA identified a total of 362,573 groups that originated from client applications. Of these, 7,715 were flagged as compromised.

Overall, our system created a total of 7,250,228 behavioral profiles. COMPA identified 966,306 messages that violate the behavioral profiles of their corresponding accounts. Finally, 400,389 messages were deleted by the time our system tried to compare these messages to their respective behavioral profiles (i.e., within an hour).

False Positives

Using the text similarity measure, COMPA identified 343,229 compromised Twitter accounts in 9,362 clusters. To analyze the accuracy of these results, we need to an-

Network & Similarity Measure	Twitter Text		Twitter URL		Facebook Text	
	Groups	**Accounts**	**Groups**	**Accounts**	**Groups**	**Accounts**
Total Number	374,920		14,548		48,586	
# Compromised	9,362	343,229	1,236	54,907	671	11,499
False Positives	4% (377)	3.6% (12,382)	5.8% (72)	3.8% (2,141)	3.3% (22)	3.6% (412)
# Bulk Applications	12,347		1,569		N/A	N/A
# Compromised Bulk Applications	1,647	178,557	251	8,254	N/A	N/A
False Positives	8.9% (146)	2.7% (4,854)	14.7% (37)	13.3% (1,101)	N/A	N/A
# Client Applications	362,573		12,979		N/A	N/A
# Compromised Client Applications	7,715	164.672	985	46,653	N/A	N/A
False Positives	3.0% (231)	4.6% (7,528)	3.5% (35)	2.2% (1,040)	N/A	N/A

Table 2. Evaluation Results for the Text (Twitter and Facebook) and URL (Twitter) Similarity measure

swer two questions: First, do we incorrectly label non-compromised accounts as compromised (false positives)? We try to answer this question in this subsection. Second, do we miss accounts that have been compromised (false negatives)? We discuss this question in the next subsection.

False positives might arise in two cases: First, legitimate users who change their habits (e.g., a user experiments with a new Twitter client) might be flagged as compromised. Second, fake accounts, specifically created for the purpose of spreading malicious content, might trigger our detection, but these are not compromised accounts. Arguably, the second source of false positives is less problematic than the first one, since the messages that are distributed by fake accounts are likely malicious. However, since social network providers need to handle compromised accounts differently from fake accounts (which can be simply deleted), we want our system to only report compromised accounts.

To address the first reason for false positives, we analyzed the groups that our similarity measures generated. First, we aggregated similar (repeated) groups into long-lasting campaigns. Two groups belong to the same campaign if all pairwise Levenshtein ratios between ten randomly-chosen messages (five messages from each group) is at least 0.8. We could aggregate 7,899 groups into 496 campaigns. We then manually analyzed a subset of the accounts present for each campaign. Additionally, 1,463 groups did not belong to any campaign, and we assessed each of these manually. During manual analysis, we opted to err on the conservative side by counting groups whose accounts contain messages written in non-Latin based languages as false positives.

In total, 377 of the 9,362 groups (4%) that COMPA flagged as containing compromised accounts could not be verified as such, and thus, constitute false positives. Note that each group consists of multiple tweets, each from a different Twitter account. Thus, the above mentioned results are equivalent to flagging 343,229 user as compromised, where 12,382 (3.6%) are false positives.

Three months after we finished our experiments, we tried to retrieve all messages that we found were indicative of compromised accounts. Only 24% were still available. Furthermore, we would expect that the majority of messages sent by legitimate accounts are persistent over time. Thus, we also tried to retrieve a random sample of 160,000 messages contained in clusters that COMPA found to be benign. 82% of these messages were still reachable. Additionally, we also tried to access 64,368 random messages that we collected during our experiments as described in the following subsection. Of these, 84% were still accessible.

This means that for 76% of the messages that COMPA identified as being sent by a compromised account, either Twitter or the user herself removed the message. However, 96.2% of the accounts that sent these tweets were still accessible. For less than one percent (0.6%) of the accounts, Twitter states that they were suspended. The remaining 3.2% return a "Not found" error upon access. These percentages are almost perfectly in line with accounts that COMPA did not flag as compromised (95.5%, 0.5%, and 4%, respectively), and a random sample of 80,000 accounts (94%, 4%, and 2%). Twitter actively suspends spam accounts on their network. Thus, these results indicate that Twitter does not consider the accounts flagged by COMPA as fake. However, the significant amount of removed messages for such accounts leads us to believe that COMPA was successful in detecting compromised accounts.

To estimate the second source of false positives, we developed a classifier based on the work of Stringhini et al. [7]. This allows us to detect accounts that were fake accounts as opposed to compromised. Stringhini's system detects fake accounts that are likely spammers, based on features related to automatically created and managed accounts (such as the ratio between the friend requests that are sent and the requests that are accepted, the fraction of messages with URLs, and how similar the messages are that a single user sends). This system proved to be effective in detecting accounts that have been specifically created to spread malicious content. However, the system does not usually detect compromised accounts. This is because such accounts usu-

ally have a long history of legitimate Twitter usage, and, hence, the few tweets that are sent out after the compromise do not affect the features their classifier relies on.

We used this classifier to analyze a random sample of 94,272 accounts that COMPA flagged as compromised. The idea is that any time an account is detected as a spammer, this is likely to be an account specifically created with a malicious intent, and not a legitimate, yet compromised account. Out of the analyzed accounts, only 152 (0.16%) were flagged as spammers. We then manually checked these accounts to verify if they were actually false positives of COMPA. 100 of these 152 accounts turned out to be compromised, and thus, true positives of COMPA. The reason for flagging them as spam accounts is that they have not been very active before getting compromised. Therefore, after the account was compromised, the spam messages had more influence on the features than the legitimate activity before the compromise. The remaining 52 accounts were not compromised but had been specifically created to spam. However, these 52 accounts were distributed over 34 clusters with an average cluster size of 30. Furthermore, no cluster consisted solely of false positives. The main reason why they were detected as behavior violations by COMPA is that they posted an update in an hour during which they had never been active before. This result underlines that the compromised accounts that COMPA reports are substantially different than the dedicated, fake accounts typically set up for spamming.

Historical information. We also investigated how the length of a user's message stream influences the quality of the behavioral profiles COMPA builds (recall that COMPA does not build a behavioral profile when a user has posted fewer than 10 messages). To this end, we calculated the probability of a false positive depending on the number of tweets that were available to calculate the behavioral profile. As Figure 2 illustrates, COMPA produces less false positives for accounts whose historical data is comprehensive. The reason for this is that the models become more accurate when more historical data is available.

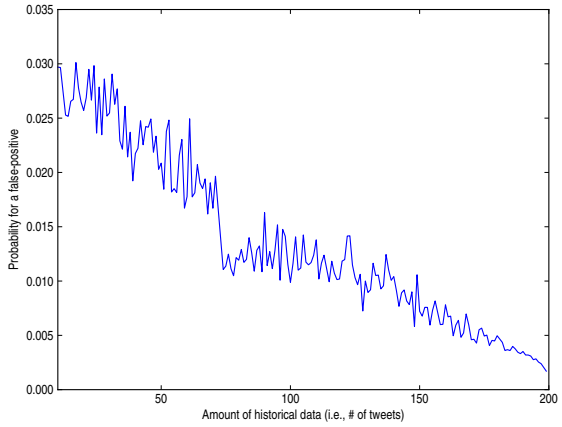

Figure 2. Probability of false positives depending on the amount of historical data on Twitter

False Negatives

To assess false negatives, we used COMPA to create 64,368 behavioral profiles for randomly selected users over a period of 44 days. To this end, every minute, COMPA retrieved the latest tweet received from the Twitter stream and built a behavioral profile for the corresponding user. 2,606 (or 4%) of these profiles violated their account's behavioral profile. 415 of these were sent by known, popular bulk applications. We manually inspected the remaining 2,191 tweets that violated their accounts' behavioral profiles (we performed the same manual analysis that was previously used to determine the ground truth for our training set). We did not find evidence of any malicious activity that COMPA missed.

In a next step, we extracted all URLs posted on Twitter during one day, and we checked them against five popular blacklists. The idea behind this is that if a URL is known to be malicious, it is likely to be posted either by a compromised or by a fake account. We extracted 2,421,648 URLs (1,956,126 of which were unique) and checked them against the *Spamhaus Domain Blacklist* [22], *Google Safebrowsing* [23], *PhishTank* [24], *Wepawet* [25], and *Exposure* [26]. We first expanded shortened URLs before checking the landing page against the blacklists. In total, 79 tweets contained links that were present in at least one blacklist (these 79 tweets contained 46 unique URLs). We ran COMPA on each of the 79 messages to see if they were actually sent by compromised accounts.

Our system flagged 33 messages as violating their user's profile. The reason COMPA did not flag these accounts in the first place is that the clusters generated by these messages were too small to be evaluated, given the API limit we mentioned before. If we did not have such a limit, COMPA would have correctly flagged them. Seven more messages contained URLs that were similar to those in the 33 messages. Even though these compromised accounts did not violate their behavioral profiles, they would have been detected by COMPA, because they would have been grouped together with other messages that were detected as violating their behavioral profiles.

Of the remaining 39 accounts that COMPA did not flag as compromised, 20 were detected as fake accounts by the classifier by Stringhini et al. [7]. We manually investigated the remaining 19 results. 18 of them contained links to popular news sites and blogs, which were mainly blacklisted by Google Safebrowsing. We think users posted legitimate links to these pages, which might have become com-

promised at a later point in time (or are false positives in Google Safebrowsing). Thus, we do not consider accounts that linked to such pages as either compromised or fake.

The remaining message linked to a phishing page, but did not violate the profile of the account that posted it. We consider this as a message by a compromised account, and, therefore, a false negative of COMPA.

6.4 Detection on Facebook

As the Facebook dataset spans almost two years we increased the *observation interval* to eight hours to cover this long timespan. Furthermore, we only evaluated the Facebook dataset with the text similarity measure to group similar messages.

Our experiments indicated that a small number of popular applications resulted in a large number of false positives. Therefore, we removed the six most popular applications, including *Mafia Wars* from our dataset. Note that these six applications resulted in groups spread over the whole dataset. Thus, we think it is appropriate for a social network administrator to white-list applications at a rate of roughly three instances per year.

In total, COMPA generated 206,876 profiles in 48,586 groups and flagged 671 groups as compromised (i.e, 11,499 compromised accounts). All flagged groups are created by bulk applications. 22 legitimate groups were incorrectly classified (i.e., 3.3% false positives) as compromised; they contained 412 (3.6%) users.

6.5 Profile Accuracy

One example to illustrate the accuracy of the behavioral profiles COMPA creates is the following. On July 4th, 2011, the Twitter account of the politics division of Fox News (@foxnewspolitics) got compromised [27]. The attackers used the account to spread wrong information about an assassination of president Obama. As this incident was limited in scope, COMPA did not observe enough messages to create a group. Therefore, we instructed COMPA to create a behavioral profile for @foxnewspolitics and have it compare the offending tweets against this profile. COMPA detected significant deviations from the created behavioral profile for all but the language models. Thus, the offending tweets posed a clear violation of the behavioral profile of the @foxnewspolitics account.

6.6 Case Studies

In this section, we describe some interesting findings about the compromised accounts detected by COMPA.

"Get more Followers" scams On Twitter, the majority of the accounts that COMPA flagged as compromised were part of multiple large-scale phishing scams that advertise "more Followers". These campaigns typically rely on a phishing website and a Twitter application. The phishing website promises more followers to a user. The victim can either get a small number of followers for free, or she can pay for a larger set of followers. Many users consider the number of their followers as a status symbol on the Twitter network, and the "base version" of the service is free. This combination seems to be an irresistible offer for many. The phishing sites requires the user to share their username and password with the website. Additionally, the user needs to give read and write access to the attacker's application. Once the victim entered her credentials and authorized the application, the application immediately posts a tweet to the victim's account to advertise itself. Subsequently, the attackers make good on their promise and use their pool of existing, compromised accounts to follow the victim's account. Of course, the victim also becomes part of the pool and will start following other users herself.

Phone numbers COMPA also detected scam campaigns that do not contain URLs. Instead, potential victims are encouraged to call a phone number. Such messages would read, for example, "Obama is giving FREE Gas Cards Worth $250! Call now-> 1 888-858-5783 (US Only)@@@." In our evaluation, 0.3% of the generated groups did not include URLs. Existing techniques, such as [13], which solely focus on URLs and their reputation, would fail to detect such campaigns.

Detecting Worms Online social networks have been repeatedly confronted with XSS worm outbreaks that rapidly infect thousands of accounts. Since the behavior of the affected accounts is expected to diverge from their usual behavioral profiles, we show in Appendix A that COMPA successfully detects such outbreaks.

7 Limitations

An attacker who is aware of COMPA has several possibilities to prevent his compromised accounts from being detected by COMPA. First, the attacker can post messages that align with the behavioral profiles of the compromised accounts. As described in Section 3, this would require the attacker to invest significant time and computational resources to gather the necessary profile information from his victims. Furthermore, social networks have mechanisms in place that prevent automated crawling, thus slowing down such data gathering endeavors.

Second, an attacker could send messages that evade our similarity measures, and thus, although such messages might violate their compromised accounts' behavioral profiles, they would not get grouped together. To counter such

evasion attempts, COMPA can be easily extended with additional and more comprehensive similarity measures. For example, it would be straight-forward to create a similarity measure that uses the landing page instead of the URLs contained in the messages to find groups of similar messages. Furthermore, more computationally expensive similarity measures, such as text shingling or edit distances for text similarity can also be implemented. Other similarity measures might leverage the way in which messages propagate along the social graph to evaluate message similarity.

8 Related Work

The popularity of social networks inspired many scientific studies in both, networking and security. Early detection systems for malicious activity on social networks focused on identifying fake accounts and spam messages [5, 6, 7] by leveraging features that are geared towards recognizing characteristics of spam accounts (e.g., the presence of URLs in messages or message similarity in user posts). Cai et al. [18] proposed a system that detects fake profiles on social networks by examining densely interconnected groups of profiles. These techniques work reasonably well, and both Twitter and Facebook rely on similar heuristics to detect fake accounts [28, 29].

In response to defense efforts by social network providers, the focus of the attackers has shifted, and a majority of the accounts carrying out malicious activities were not created for this purpose, but started as legitimate accounts that were compromised [3, 4]. Since these accounts do not show a consistent behavior, previous systems will fail to recognize them as malicious. Grier et al. [4] studied the behavior of compromised accounts on Twitter by entering the credentials of an account they controlled on a phishing campaign site. This approach does not scale as it requires identifying and joining each new phishing campaign. Also, this approach is limited to phishing campaigns. Gao et al. [10] developed a clustering approach to detect spam wall posts on Facebook. They also attempted to determine whether an account that sent a spam post was compromised. To this end, the authors look at the wall post history of spam accounts. However, the classification is very simple. When an account received a benign wall post from one of their connections (friends), they automatically considered that account as being legitimate but compromised. The problem with this technique is that previous work showed that spam victims occasionally send messages to these spam accounts [7]. This would cause their approach to detect legitimate accounts as compromised. Moreover, the system needs to know whether an account has sent spam before it can classify it as fake or compromised. Our system, on the other hand, detects compromised accounts also when they are not involved in spam campaigns. As an improvement to these techniques, Gao et al. [10] proposed a system that groups similar messages posted on social networks together, and makes a decision about the maliciousness of the messages based on features of the message cluster. Although this system can detect compromised accounts, as well as fake ones, their approach is focused on detecting accounts that spread URLs through their messages, and, therefore, is not as generic as COMPA.

Thomas et al. [13] built Monarch to detect malicious messages on social networks based on URLs that link to malicious sites. By relying only on URLs, Monarch misses other types of malicious messages. For example, the scams based on phone numbers that COMPA detected would not be detected. It also would not detect a XSS worm spreading without a URL, as well as a new, emerging kind of spam that includes incomplete links in the tweet (e.g., a missing `http://`). These spam messages ask users to copy and paste a fragmented URL in the browser address bar [30], where the URL is automatically reassembled. Lee et al. [12] proposed WARNINGBIRD, a system that detects spam links posted on Twitter by analyzing the characteristics of HTTP redirection chains that lead to a final spam page.

Xu et al. [31] present a system that, by monitoring a small number of nodes, detects worms propagating on social networks. This paper does not directly address the problem of compromised accounts, but could detect large-scale infections such as *koobface* [2].

Yang et al. [17] studied new Twitter spammers that act in a stealthy way to avoid detection. In their system, they use advanced features such as the topology of the network that surrounds the spammer. They do not try to distinguish compromised from spam accounts.

9 Conclusions

In this paper, we presented a novel approach to detect compromised accounts in social networks. More precisely, we developed statistical models to characterize the behavior of social network users, and we used anomaly detection techniques to identify sudden changes in their behavior. We developed COMPA, a prototype tool that implements this approach, and we applied it to a large stream of messages. The results show that our approach reliably detects compromised accounts, even though we do not have full visibility of every message exchanged on Facebook and Twitter.

Acknowledgements

This work was supported by the Office of Naval Research (ONR) under Grant N000140911042, the Army Research Office (ARO) under grant W911NF0910553, the National Science Foundation (NSF) under grants CNS-0845559 and CNS-0905537, and Secure Business Austria.

References

[1] Harris Interactive Public Relations Research, "A Study of Social Networks Scams," 2008.

[2] J. Baltazar, J. Costoya, and R. Flores, "KOOBFACE: The Largest Web 2.0 Botnet Explained," 2009.

[3] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Zhao, "Detecting and Characterizing Social Spam Campaigns," in *Internet Measurement Conference (IMC)*, 2010.

[4] C. Grier, K. Thomas, V. Paxson, and M. Zhang, "@spam: the underground on 140 characters or less," in *ACM Conference on Computer and Communications Security (CCS)*, 2010.

[5] F. Benvenuto, G. Magno, T. Rodrigues, and V. Almeida, "Detecting Spammers on Twitter," in *Conference on Email and Anti-Spam (CEAS)*, 2010.

[6] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: social honeypots + machine learning," in *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2010.

[7] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting Spammers on Social Networks," in *Annual Computer Security Applications Conference (ACSAC)*, 2010.

[8] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Your Botnet is My Botnet: Analysis of a Botnet Takeover," in *ACM Conference on Computer and Communications Security (CCS)*, 2009.

[9] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda, "All Your Contacts Are Belong to Us: Automated Identity Theft Attacks on Social Networks," in *Wold Wide Web Conference (WWW)*, 2009.

[10] H. Gao, Y. Chen, K. Lee, D. Palsetia, and A. Choudhary, "Towards Online Spam Filtering in Social Networks," in *Symposium on Network and Distributed System Security (NDSS)*, 2012.

[11] "foursquare," http://foursquare.com.

[12] S. Lee and J. Kim, "WarningBird: Detecting Suspicious URLs in Twitter Stream," in *Symposium on Network and Distributed System Security (NDSS)*, 2012.

[13] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and Evaluation of a Real-Time URL Spam Filtering Service," in *IEEE Symposium on Security and Privacy*, 2011.

[14] "Oauth community site," http://oauth.net.

[15] W. B. Cavnar and J. M. Trenkle, "N-gram-based text categorization," in *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, 1994, pp. 161–175.

[16] J. C. Platt, "Fast Training of Support Vector Machines Using Sequential Minimal Optimization," in *Advances in Kernel Methods - Support Vector Learning*, 1998.

[17] C. Yang, R. Harkreader, and G. Gu, "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.

[18] Z. Cai and C. Jermaine, "The Latent Community Model for Detecting Sybils in Social Networks," in *Symposium on Network and Distributed System Security (NDSS)*, 2012.

[19] J. Song, S. Lee, and J. Kim, "Spam Filtering in Twitter using Sender-Receiver Relationship," in *Symposium on Recent Advances in Intrusion Detection (RAID)*, 2011.

[20] "Surbl," http://www.surbl.org.

[21] "Weka - data mining open source program," http://www.cs.waikato.ac.nz/ml/weka/.

[22] "Spamhaus dbl," http://www.spamhaus.org.

[23] "Google safebrowsing," http://code.google.com/apis/safebrowsing/.

[24] "Phishtank," http://www.phishtank.com.

[25] "Wepawet," http://wepawet.iseclab.org.

[26] "Exposure," http://exposure.iseclab.org/.

[27] "Fox news's hacked twitter feed declares obama dead," http://www.guardian.co.uk/news/blog/2011/jul/04/fox-news-hacked-twitter-obama-dead, 2011.

[28] C. Ghiossi, "Explaining Facebook's Spam Prevention Systems," http://blog.facebook.com/blog.php?post=403200567130, 2010.

[29] Twitter, "The twitter rules," http://support.twitter.com/entries/18311-the-twitter-rules, 2010.

[30] F-Secure, "The increasingly shapeshifting web," http://www.f-secure.com/weblog/archives/00002143.html.

[31] W. Xu, F. Zhang, and S. Zhu, "Toward worm detection in online social networks," in *Annual Computer Security Applications Conference (ACSAC)*, 2010.

[32] "Nielsen," http://blog.nielsen.com.

A Detecting Worms

Twitter has been affected by multiple worm outbreaks. For example, in September 2010 a XSS worm exploited a vulnerability in the way in which Twitter parsed URLs in tweets. More specifically, if a URL contained an "@" symbol, Twitter would interpret everything following that character as JavaScript. Therefore, a user who hovered her mouse over a tweet containing a URL similar to *http://x.xx/@"onmouseover="alert(1)* would execute the JavaScript event handler in her browser. Of course, the real worm used JavaScript that would self propagate the worm instead of the `alert` statement. Obviously, posting the tweet that contained the body of the worm happened without the user's consent, and, therefore, we have to consider such accounts as compromised. Note that the URL preceding the @ sign was irrelevant for the attack. Therefore, existing detection approaches that examine the maliciousness of URLs would fail to detect this XSS worm attack, as the attacker could chose any benign domain (e.g., http://www.google.com).

To evaluate whether COMPA is capable of detecting worm outbreaks, we simulated the worm outbreak on real Twitter data. That is, we chose a random message S_0 of a random user U_0 on the Twitter network. We assumed that the worm would propagate from user *A* to user *B* iff user *B* follows user *A*, and user *B* was active on Twitter within a time window *T* around the point in time when user *A* posts the offending message. Due to the lack of detailed usage information, we determine the activity of a user by observing when they tweet. Thus, a user is deemed active *T*/2 before and after she posted any status updates through the Twitter web interface. Note that this definition of activity (i.e., a user is only deemed active when she is posting) is conservative, as users are often browsing Twitter or reading other people's tweets, even if they do not post at the same time. Furthermore, the worm only propagates itself if the tweet that user *B* sent was posted through the Twitter web site. That is, alternative clients are assumed not to contain the same vulnerability in their URL parsing routines. The XSS worm we are simulating is aggressive in that it spreads as soon as the user hovers the mouse over the tweet. We assume that if a user is active, she will hover her mouse over the tweet, and thus, get infected. For every propagation step, we record the IDs of users *A* and *B*, as well as the ID of the tweet that was used to determine that user *B* is active (i.e., the tweet user *B* sent within the time window *T*). According to [32], web users spend roughly 10 minutes per day on social networks. Thus, we assumed a value of 10 minutes for *T* in our simulation.

Subsequently, we downloaded the timelines of the users infected by the simulated worm. Then, we substituted the tweets that were responsible for the worm propagation with a copy of the XSS worm. Finally, we ran COMPA on these timelines. Although the way we simulated the worm outbreak means that the timing and source models are drawn from real information (i.e., we only substituted the text of the tweet), COMPA was able to successfully detect the outbreak and the compromised accounts after the worm spread to 2,256 accounts in 20 minutes. This means that the "worm group" contained enough tweets that violated their respective users' behavioral profiles. It turns out that our propagation strategy was chosen conservatively as news reports[2] of previous Twitter worms report of 40,000 infected accounts within 10 minutes. Thus, assuming the distribution of profile violations is similar for such aggressive worms, COMPA would detect such a large scale outbreak even faster.

[2] http://eu.techcrunch.com/2010/09/21/warning-mouseover-tweets-security-flaw-is-wreaking-havoc-on-twitter/