Characterizing and Detecting Malicious Accounts in Privacy-Centric Mobile Social Networks: A Case Study

Zenghua Xia
Tsinghua University
xiazh16@mails.tsinghua.edu.cn

Chang Liu
Citadel Securities
liuchang2005acm@gmail.com

Neil Zhenqiang Gong
Duke University
neilz.gong@gmail.com

Qi Li
Tsinghua University
qli01@tsinghua.edu.cn

Yong Cui
Tsinghua University
cuiyong@tsinghua.edu.cn

Dawn Song
UC Berkeley
dawnsong@cs.berkeley.edu

ABSTRACT

Malicious accounts are one of the biggest threats to the security and privacy of online social networks (OSNs). In this work, we study a new type of OSN, called *privacy-centric mobile social network* (PC-MSN), such as KakaoTalk and LINE, which has attracted billions of users recently. The design of PC-MSN is inspired to protect their users' privacy from strangers: (1) a stranger is not easy to send a friend request to a user who does not want to make friends with strangers; and (2) strangers cannot view a user's post. Such a design mitigates the security issue of malicious accounts. At the same time, it also brings the battleground between attackers and defenders to an earlier stage, i.e., making friendship, than the one studied in previous works. Also, previous defense proposals mostly rely on certain assumptions on the attacker, which may not be robust in the new PC-MSNs. As a result, previous malicious accounts detection approaches are less effective on a PC-MSN.

To mitigate this issue, we study the patterns in friend requests to distinguish malicious accounts, and perform a systematic study over 1 million labeled data from WLink, a real PC-MSN with billions of users, to confirm our hypothesis. Based on the results, we propose dozens of new features and leverage machine learning to detect malicious accounts. We evaluate our method and compare it with existing methods, and the results show that our method achieves a precision of 99.5% and a recall of 98.4%, which significantly outperform previous state-of-the-art methods. Importantly, we qualitatively analyze the robustness of the designed features, and our evaluation shows that using only robust features can achieve the same level of performance as using all features. WLink has deployed our detection method. Our method can detect 0.59 million malicious accounts daily, which is 6× higher than the previous deployment on WLink, with a precision of over 90%.

CCS CONCEPTS

• **Security and privacy → Social network security and privacy**;

KEYWORDS

Malicious accounts detection, online social networks, friend request, neural networks

ACM Reference Format:
Zenghua Xia, Chang Liu, Neil Zhenqiang Gong, Qi Li, Yong Cui, and Dawn Song. 2019. Characterizing and Detecting Malicious Accounts in Privacy-Centric Mobile Social Networks: A Case Study. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330702

1 INTRODUCTION

Online social networks (OSNs) are known to be vulnerable to malicious accounts. An attacker can automatically maintain a large number of malicious accounts and use them to perform various malicious activities including, but not limited to, disrupting the presidential election and stock markets via spreading fake news [32, 33], distributing malware and phishing URLs [45], as well as stealing users' private data [6]. Therefore, detecting malicious accounts in OSNs is an urgent and basic security research problem.

Recently, a new class of social networks, namely *privacy-centric mobile social networks* (PC-MSNs), has attracted billions of users. PC-MSNs, such as LINE [3] and KakaoTalk [2], are initially inspired to prevent many social engineering attacks. For example, the attackers may first probe the victim's activities posted on the social network, and then make a call to the victim to pretend to be her friends while increasing his trustworthiness by making stories leveraging the social activities of the victim. To mitigate such kinds of attacks, PC-MSNs are designed to enforce their user privacy: (1) *strangers* are not easy to make a friend with a benign user unless the later intentionally approve it; and (2) a user's posts on the network are not visible to others who are not a friend of the user. PC-MSNs' fast adoption is partially attributed to such a *privacy-centric* design.

Clearly, detecting malicious accounts on PC-MSNs is still a crucial problem. In this work, we study the malicious accounts problem on WLink[1] , a large PC-MSN with over a billion monthly active users. One straightforward idea is to apply the methods proposed for malicious OSN accounts detection, which has been studied over a decade [11, 14, 15, 17, 18, 22, 23, 27, 41–44, 46, 55]. Unfortunately, we observe that many such approaches are no longer effective for PC-MSNs (see Section 5.2).

[1]Due to the confidentiality agreement, we do not disclose the real name, but use the pseudonym WLink instead.

We study why existing approaches are not effective. One of the main reason is due to their privacy-centric design. In fact, the privacy policy of WLink incurs an additional barrier between the attacker and the victims. Different from existing OSNs, the attacker needs to become a friend of the victim first before performing malicious activities. At the same time, WLink provides limited ways for one user to send friend requests to another. For example, one way to send a friend request is through scanning the QR code, which mostly requires the two users to be physically close to each other, and thus it is less likely for one user to add a stranger as her friend. Therefore, WLink essentially moves the battleground between attackers and defenders to the field of making friendship, a.k.a., sending friend requests, which is a much earlier stage than the attack scenarios studied in the literature [12, 43], which motivates the design of previous yet less effective detection proposals.

On the other hand, some unsupervised detection proposals [12, 44] rely on assumptions on the attacker in their design. Although these works argue that the used assumptions are *robust*, i.e., the attacker needs to incur a high cost to violate them, we find that the malicious accounts on WLink do not meet these assumptions, and thus they can *evade* such detection algorithms. Therefore, another important question that has not been carefully understood is how to design *robust* features to defend against evasion attackers, especially in our scenario to build the detection algorithm using *supervised learning*.

In this work, we are the first to (1) study malicious accounts on a large PC-MSN, i.e., WLink, with respect to the main battleground, i.e., friend requests; and (2) discuss how to design robust features against evasion attackers for the supervised malicious accounts detection problem. To examine our hypothesis, we conduct a measurement study over a labeled dataset of around 1 million benign and malicious accounts collected from WLink. We observe that many attributes related to the friend requests can effectively distinguish malicious accounts from benign ones. For example, malicious accounts tend to send significantly more friend requests; malicious accounts are significantly more likely to pretend to be a female and send friend requests to male users, while benign accounts are significantly more likely to send friend requests to female users; and malicious accounts tend to use a diverse set of request messages while benign accounts' are similar. Based on these observations, we propose a rich set of features and quantitatively analyze their robustness.

Note that existing works [10, 25] have studied friend requests and its application to malicious account detection. However, these works only considered each friend request as a directed graph edge on the social network graph and did not examine richer information contained in a friend request. The novelty of this work, in contrast, is to have a better understanding of the information contained in friend requests without considering the graphical structure, and its application to the malicious accounts prevention problem. More importantly, this work is the first to analyze which features are more robust against evasion attackers.

Based on the proposed features, we employ a deep neural network (DNN) to convert the textual messages in a friend request into features, and use a Random Forest [8] to build a malicious accounts detection prototype, called Realguard, and compare it against several previous state-of-the-art proposes, such as Stringhini et al. [43], EvilCohort [44] and SynchroTrap [12]. We observe that our approach can achieve an F1-score of over 98.9%, while existing approaches' scores range from 50.7% to 88.8%. More importantly, our evaluation reveals that using only *robust* features can achieve comparable performance to using all features; this result also sheds light on employing robust features against evasion attackers to detect malicious accounts on an OSN. We also deploy Realguard in WLink's production pipeline, and Realguard is able to analyze around 1 billion friend requests sent from millions of users daily and detects 0.59 million malicious accounts per day, which is 6× more than the previous deployment in WLink. Through manual inspection, it is confirmed that Realguard can achieve over 90% precision.

We summarize our contributions as follows:

- This is the first work studying malicious accounts on a large PC-MSN, i.e., WLink;
- We perform a large-scale measurement study to characterize malicious accounts in WLink;
- Based on the measurement study, we propose dozens of new features and leverage supervised machine learning to detect malicious accounts;
- This is the first work to analyze the robustness of each proposed feature with respect to a potential evasion attacker who tries to evade the malicious account detector.
- We implement a prototype, called Realguard, of our method and compare it with the state-of-the-art methods using the labeled dataset from WLink. Our results show that our method can achieve a precision of 99.5% and a recall of 98.4%, which significantly outperform the baseline methods;
- We deploy Realguard on WLink's production line to process billions of friend requests sent from millions of accounts daily. Our system can detect 6× more malicious accounts per day than the previous deployment with a precision over 90%.

2 PRIVACY-CENTRIC MOBILE SOCIAL NETWORKS

In this section, we will describe the key design and functionality of PC-MSN, using WLink as an example. Other networks such as LINE and KakaoTalk share all or partial similar properties. WLink is designed to have a strict policy to protect users' privacy from strangers, whom the user may not know in person. Such privacy policies include but not limited to that (1) a stranger cannot easily add a user as a friend; and (2) the social activities of a user, such as posts, cannot be viewed by strangers. Such a design makes attackers, who are mostly strangers to the victim, harder to spread malicious content. In the following, we will first detail the two unique designs of WLink; and then we will briefly discuss the characteristics of the malicious accounts in such a network.

Limited channel usage for sending friend requests: WLink provides limited channel usage for a user to find other users and send them friend requests. Different from other OSNs, such as Renren [4] or Facebook, WLink does not allow searching for friends with their names, which makes it more difficult to make friends since WLink ID is much more private and hard to know for strangers; neither does WLink suggest friends to a user to add. On WLink, a user can send a friend request to another user using its ID, phone number, and *name card*, which is used by a user to share information between two of her friends. A WLink user can also send a friend request to another by scanning the latter's QR code. In addition, when a user is invited to join a group, the user can send friend requests to the group members. These channels mostly leverage mobile devices' features to require the receivers of friend requests to have personal contact with the senders in reality. Thus, these channels are harder for strangers to leverage.

A user can also send friend requests to other users based on certain services in WLink. For instance, a user can use location-based service (LBS) to find other users that are close and also use the LBS; then the user can send them friend requests. A user can pick up a message in the *RandomMsg* service and sends a friend request to the sender of the message. Finally, when two users use the "match" service at the same time, a user can obtain the other user's basic account information and send a friend request. These channels enable strangers to send friend requests to the receiver; however, the receiver requires to opt-in to receive friend requests through these channels. This means that if a user chooses to opt-out, a stranger's friend requests through these channels will be automatically rejected.

Therefore, such a design can mitigate the friend request issue from strangers on principle. In practice, however, from Section 3, we will observe that a high volume of malicious accounts still send friend requests through QRCode and RandomMsg.

Restricted content sharing: WLink has a more restricted content sharing policy than Twitter and Facebook. For instance, on Twitter, a user can use *mentions* to target any specific user, e.g., a user can post a tweet *"@XYZ look at this cool website!"*, which will be shown to the user *XYZ* even if *XYZ* is not a follower or followee of the user. Moreover, on Twitter, a user can post tweets with a certain *hashtag*, which indicates a certain topic. When a large number of users (e.g., malicious users) post tweets with the same hashtag, the topic may become a trending topic, and the tweets with this hashtag could be propagated to a large number of benign users. On Facebook, when a user sets its *timeline* to be public, its posts and comments to the posts are visible to all other users. Moreover, on Facebook, a user can post comments on other users' timeline even if they are not on the user's friend list. Therefore, malicious accounts can propagate malicious content to benign users without establishing connections with them on Twitter and Facebook [28, 31].

In WLink, two users can interact with each other via posts, comments, private messages, and group messages. Specifically, a normal user's posts are only visible to and can be commented by the user's friends. Suppose user A adds a comment to her friend B's post. The comment is only visible to the common friends of user A and user B. The comments of user A cannot be re-shared by user B unless user B posts a new comment with the same content. Two users can exchange private messages only if they are friends. A user can be invited by her friend to join a group and exchange messages in the group. All these content propagation requires users to be connected, e.g., a user must be connected to at least one group member in group messages. These mechanisms make information propagate mostly among friends.

Another way to propagate content is via sending friend requests which allow a *request message* (up to 120 bytes) attached. The receiver of a friend request will be shown with the request message. However, WLink mostly requires its users to send friends to only whom they may know each other offline.

We note that WLink provides a service called *RandomMsg*, in which a user can send out messages that could be received by some random users who are also using the service. However, WLink only allows a very limited number of random messages, and they can only be received by users who are using the service as well.

Therefore, to propagate malicious content at scale in WLink, malicious users often need to actively send a large number of friend requests to benign users with a goal to either accumulate many "friends" or propagate malicious content via the request messages.

Due to the unique designs on restricted content sharing and limited channel usage for sending friend requests, malicious accounts may have abnormal friend request patterns. Therefore, we focus on leveraging friend requests to characterize and detect malicious accounts in WLink.

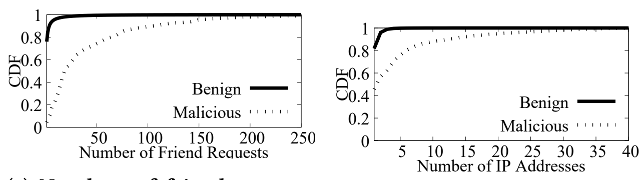
Malicious account activities in WLink. In this work, we are mainly focusing on the malicious accounts which are used to conduct illegal activities, such as spreading pornography, spam, or scam, prostitution, producing fake clicks, etc. Due to the above design, malicious accounts on WLink need to become a friend of the victims to widely spread malicious content through posting. On the other hand, if a malicious account wants to make "cold call" to spread malicious content, the only effective tool is through *friend request messages*. Such a property brings the fight between attackers and malicious accounts detectors earlier to the battleground of friend requests. This is quite different than attackers studied in previous works, which focus mainly on malicious activities, such as posting and sharing after a friendship has been made. Note that some existing works [10, 25] also studies friend requests on social networks such as Renren. However, Renren provides a search functionality to allow malicious accounts to send any other user a friend request. Therefore, making friends is not a main barrier for the attackers as in these networks.

3 CHARACTERIZING MALICIOUS ACCOUNTS

We perform a systematic measurement study to characterize the friend request patterns of malicious accounts and compare them with those of benign accounts in WLink. Throughout the analysis, we focus not only on what features are indicative for predicting malicious accounts but also whether these features are *robust*. That is, we say a feature is robust, if (1) the attacker cannot change the value of the feature; (2) the attacker needs to degrade the utility significantly to do so; or (3) the attacker needs to incur a significant cost to manipulate feature values.

3.1 Dataset

We obtained a real-world dataset with 785,710 labeled benign accounts and 314,493 labeled malicious accounts from WLink. The security team of WLink took a conservative labeling process to obtain these labels. Therefore, the labels can be considered relatively accurate. Specifically, an account may not have a label when the



(a) Numbers of friend requests sent **(b) Unique IP addresses used**

Figure 1: CDF of distributions of benign and malicious friend requests.

security team is unsure, and such accounts are not included in our dataset. For each user, the dataset contains the user's following data: (1) profile information; (2) aggregated social activities, e.g., total number of posts and comments; and (3) friend requests sent within one day. In total, there are 2.54 million and 13.38 million friend requests sent from benign and malicious accounts, respectively. A friend request contains various fields including Sender, Receiver, Timestamp, IP, Channel and Message, which are summarized in Table 3 in reproducibility materials.

Ethics. WLink requires its users to agree to the Terms of Use, which authorizes WLink to collect the information used in this study for security auditing and we work on anonymized and aggregate data and don't attempt to deanonymize those users.

3.2 Measuring Friend Requests

We first analyze different attributes to understand how attackers use their malicious accounts to send friend requests, and further, to distinguish malicious from benign users. We refer to all friend requests sent from benign accounts as *benign friend requests*, and all friend requests from malicious accounts as *malicious friend requests*. In particular, we study (1) friend request volumes; (2) gender of the sender and receiver; (3) IP addresses used to send the friend requests; (4) channels through which the senders know the receivers; and (5) temporal patterns of the friend requests.

Number of Friend Requests Sent. In Figure 1a, we plot the Cumulative Distribution Function (CDF) of the distributions of friend request amounts sent from benign users versus malicious ones in our dataset. We observe a significant difference between the two: over 90% of benign accounts sent at most 5 friend requests within one day, while this percentage for malicious accounts is less than 10%. The reason is that WLink has a restricted content sharing policy. Malicious accounts need to aggressively send friend requests to other users to either accumulate more friends or spread malicious content via request messages. Interestingly, such a phenomenon has also been observed in traditional OSNs, such as Twitter [21] and RenRen [54]. However, previous studies [21, 54] did not provide a more fine-grained characterization of friend requests.

Since sending friend request messages is the main tool for a malicious account to make "cold-calls", lowering the volume of friend requests will also degrade the attacker's utility. Therefore, this attribute can be considered robust.

IP addresses. Intuitively, benign users tend to stay in a relatively small number of places, and thus send their friend requests from a small set of IP addresses. In contrast, malicious users may tend to use a diverse set of IP addresses (e.g., via VPN or Tor) to evade IP-based blacklisting and detection. Figure 1b shows the CDF of the distributions of the number of unique IP addresses from which benign and malicious friend requests are sent. The results confirm the above intuition, i.e., malicious accounts tend to use a more diverse set of IP addresses. For instance, around 33% of malicious friend requests used more than 3 IP addresses, while only 1% of benign accounts used more than 3 IP addresses. We consider attributes related to IP address as robust since an attacker needs to maintain many IP addresses to evade IP-based blacklisting and detection, while benign users do not have such a need.

S \ R	Female	Male
Female	34.3%	13.6%
Male	37.2%	14.9%

(a) benign accounts

S \ R	Female	Male
Female	30.2%	63.2%
Male	2.3%	4.3%

(b) malicious accounts

Table 1: Distribution of friend requests based on the Sender's and Receiver's genders.

Gender. In WLink, each user can set the gender in his/her account profile. Thus, each friend request can be classified into one of the four classes based on their sender's and receiver's genders (F→F, F→M, M→F, M→M). We calculate the distributions of these four classes for both benign and malicious friend requests, and Table 1 shows the results.

We observe that among the benign friend requests, there are about the same percentage of requests sent from male users (52.1%) and female users (47.9%). Moreover, no matter whether the sender is male or female, a benign user is significantly more likely to send friend requests to female users. Specifically, benign friend requests sent to females are 2.5× more than those sent to males.

However, we observe significantly different patterns for malicious accounts. In particular, over 93% of malicious friend requests are sent from females. Moreover, malicious friend requests are around twice more likely to be sent to a male user (e.g., 63.2% from a female) than a female user (e.g., 30.2% from a female).

We speculate the reason may be that many of these malicious accounts are used to spread pornographic information. Specifically, WLink allows a user to arbitrarily set its gender and profile picture. To attract more victims (mostly males), setting their gender to be female while sending friend requests to males can increase the chance of the friend requests being accepted. In addition, we manually investigated a small sample of malicious accounts. We observed that a majority of them use beautiful women's pictures as their profile photos, which further supports our speculation.

Since forging the gender is one of the main tools for a malicious account to attract a victim through friend requests, we think arbitrarily changing the value of this feature will degrade the attacker's utility. Therefore, we consider the gender attribute robust.

Channels. We observe that malicious users leverage a more diverse set of channels to send friend requests. For instance, above 99% of benign accounts sent their friend requests through *at most 3 channels*, while 20% of malicious accounts use *more than 3 channels*. The reason is that malicious users aim to increase the likelihood of accumulating friends via a diverse set of channels.

Figure 2 shows the distributions of the benign and malicious friend requests with respect to the 8 channels shown in Table 4.

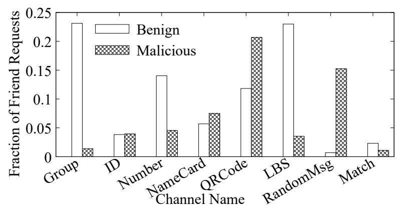


Figure 2: The distributions of friend requests with respect to channels.

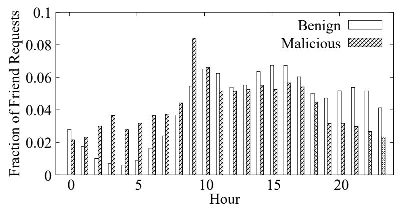


Figure 3: Hourly distributions of friend requests.

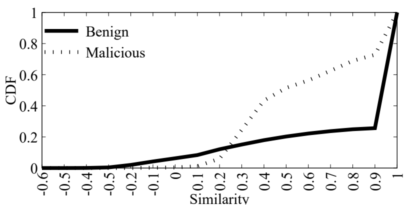


Figure 4: Similarity of request messages.

We observe that the top-3 most popular channels for benign friend requests are LBS, Group, and phone number, which in total account for around 60% of benign friend requests. Benign users often find friends through their phone numbers, e.g., WLink will recommend a user's phone contacts who also use WLink to the user. Users form groups because they have common interests or certain social connections (e.g., classmates) in the physical world, and thus benign users are more likely to send friend requests to group members.

However, we observe a significantly different pattern for malicious accounts. Specifically, the top-3 most popular channels for malicious friend requests are QRCode, RandomMsg, and NameCard, which in total account for around 43% of malicious friend requests. Our observations indicate that many malicious users pick up messages from the *RandomMsg* service and send friend requests to the senders of the messages. Interestingly, around 20% of malicious friend requests are sent via QRCode. Note that WLink does not verify this channel information on the server side, thus the attackers can manipulate the channels of their friend requests, which may be the reason why so many malicious friend requests are sent via QRCode. Therefore, the related attributes are not robust.

Time. We then study the time when the friend requests are sent. We present the hourly distributions with respect to benign and malicious requests in Figure 3. One interesting phenomenon is that the percentage of malicious requests during 1:00 am and 8:00 am is much higher than benign ones. We refer to this period as *sleeping hours*. We attribute this phenomenon to that some attackers may automate the process to send friend requests, and thus these portion of the attackers are not affected by the sleeping hour.

However, since such automated malicious friend requests are easier to detect, we find that a large number of attackers may not simply use automated attacks. In fact, we observe that over half of the malicious accounts did not send any friend requests during sleeping hours. We speculate the reason could be that those malicious accounts are mimicking benign users' temporal behavior to avoid detection. This also contributes to the phenomenon that more malicious friend requests are sent during daytime. We do not consider timing-related attributes robust, since an attacker can easily manipulate their active time to pretend to be benign.

3.3 Request Messages

The request message along with a friend request is the main information for the receiver to decide whether or not accepting the request. Therefore, different from traditional OSN, a PC-MSN account needs to carefully design the request messages along with their friend requests. Note that, normally, this message is only used to indicate the identity of the user sending the request. In this section, we present both a qualitative study and a quantitative study to understand the difference of the request messages between benign and malicious accounts.

Qualitative study. In WLink, a default template is provided for the request message field: "`I am [Name]`", where `[Name]` indicates the nickname of the sender. We note that benign users may replace the nicknames with their real names. We observe that a majority of the benign friend requests simply use such a default template generated message, or add a little more information about the sender's identity.

While such a message can be easily verified by the receiver to approve the friend request if the receiver knows the sender, an attacker is harder to get approved since he does not have a real-world connection with most receivers. Thus, we observe that malicious accounts tend to send a variety of vague request messages. Therefore, such request messages can be used as robust attributes in the sense that the attacker needs to pay a higher cost with a lower utility to construct valid request messages.

Quantitative study. Next, we study the similarity of the request messages sent by the same account. There have been many approaches to measure the similarity between two textual messages [13, 36, 43]. In this work, we employ the word2vec approach [40] to convert a textual message into a numerical vector, called *embedding*. Each word can be represented as a vector. A sentence can also be represented as a vector by simply concatenating the vector representations of words. If we set the maximum number of words, the vector representation of each sentence is ensured to be of the same dimension. Then, the similarity of two textual messages can be measured by the cosine distance between the two embeddings. For each account, we can compute the average of the pair-wise similarities of the request messages sent by the account. We plot the distributions of the similarities for benign and malicious accounts in Figure 4. We observe a significant difference between the two distributions. In particular, over 75% of the benign accounts have a similarity higher than 0.9, while this percentage of malicious ones is less than 25%. This shows that benign accounts tend to use a uniform format as their request messages, while malicious accounts may choose their request messages from a variety of templates. Note that in previous studies, the opposite phenomenon was shown overall posts published by the same account. This shows that request messages are used in a different way than posts.

Robustness. We consider the message contained in a friend request robust, since this is the only tool for a malicious account to make "cold-calls" to spread malicious content. However, we do not

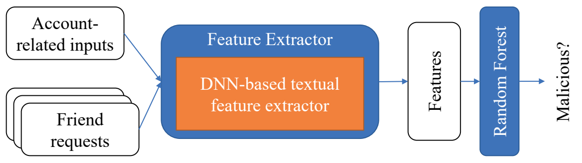


Figure 5: Malicious accounts detection workflow

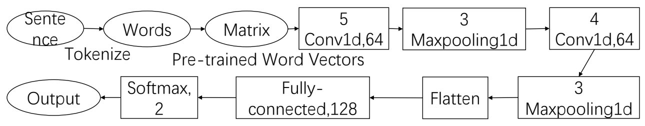


Figure 6: CNN architecture for textual feature generation.

consider the similarities among friend request messages from the same sender as a robust feature. Although we observe a pattern in Figure 4, an attacker can easily eliminate this pattern by making all messages sent from the same account similar to each other.

4 MACHINE LEARNING-BASED MALICIOUS ACCOUNTS DETECTION

In this section, we present our machine learning-based algorithm for malicious accounts detection. Our solution consists of two components: a feature extractor and a classifier. In particular, we first design a set of effective features motivated by our measurement studies in the last section, and then design a feature extractor to generate a set of features from the raw attribute values. In the end, we employ the random forest algorithm [8] to train a classifier.

The malicious accounts detection algorithm. The overall algorithm is presented in Figure 5. Both account-related inputs and friend requests are processed by the feature extractor to construct a feature vector. In particular, the feature extractor has a deep neural network (DNN) module to convert the textual input, i.e., request messages, into a numeric feature. Then a random forest is used to classify each feature vector into a label indicating whether the corresponding account is benign or malicious.

Features. We designed the features motivated by our measurement study. The full details can be found in the appendix for reproducibility. We highlight some characteristics of our features which are mostly different from the ones used in previous studies. Our features can be classified into *non-textual* ones and *textual* ones. The former includes basic statistical information about both account-related information (e.g., number of bound bank cards) and friend requests (e.g., number of unique channels used to send friend requests). The latter are generated from the request messages, which are rarely used in previous studies.

DNN-based feature extractor. Now we detail the DNN-based feature extractor. Given a request message m, we want to learn a function f, so that $f(m) \in [0, 1]$ indicates the probability of m being a good request message.

We model f as a deep neural network. Given m is a sequence of characters with a variable length, we can model f as either a *recurrent neural network* (RNN) [38], or a *convolutional neural network* (CNN) [35]. We observe that both methods produce similar accuracy, but CNN delivers better runtime performance. Thus we choose to use a CNN to model f. The concrete neural network architecture is provided in Figure 6. In particular, in the second step, we use the same word2vec model to convert a message into a matrix as described in Section 3.3. In the end, the softmax output is a 2-dimensional vector, where its first dimension's value is outputted as $f(m)$. During prediction, once the message m is mapped into $f(m)$, SumP is computed as the summation of $f(m)$ for all messages.The other five aggregated metrics are computed similarly.

Feature robustness. As we mentioned above, we consider a feature to be robust, if (1) the attacker cannot manipulate the value of the feature; (2) the attacker needs to incur significant cost to do so; or (3) the attacker's utility will be degraded in doing so. Most of the robust friend requests-related features have been discussed in Section 3.

The only robust account-related features are gender and the number of bound bank cards. Since banks require verifying the real identity of their customers to open an account and to get a card, it will incur a significant cost for a malicious account to bind many bank cards, and will also increase the chance to unveil the real identity of the attacker. Therefore, we think this feature is robust.

5 EVALUATION

In this section, we evaluate our approach. We will first explain the experimental setup and then present the results. In the end, we show our real-world evaluation by deploying our prototype in the WLink pipeline.

5.1 Experimental Setup

We use two datasets. The first one is the same dataset used in our measurement study: 785,710 benign accounts and 314,493 malicious accounts that have been detected and verified by WLink's security department. We use it for most experiments. We denote the difference between the date an account is registered and the date its friend requests are recorded in our dataset as account age. Our dataset consists of accounts of various account ages varying from days to thousands of days. In particular, we split the data into training, validation and testing set, accounting for 50%, 10%, and 40% respectively. The second dataset consists of more than one billion records and about 20 million unlabeled accounts with a time span of one day, which is collected three months after the first dataset. We use this dataset to evaluate the effectiveness of our model when deployed in a real-world production line.

The implementation details can be found in the appendix, which also explains the three state-of-the-art solutions, SynchroTrap [12], Evilcohort [44], and Stringhini et al. [43], used for comparison.

5.2 Results

5.2.1 Compare with previous approaches. We first evaluate previous approaches as well as Realguard on the real dataset collected from WLink. Figure 7 presents the results. We can observe that previous approaches are not well-performing on WLink's data. In particular, we observe that none of the previous approaches can achieve an F1 score higher than 90%. Especially, the Evilcohort approach's recall is as low as 35%, which shows that it cannot effectively detect malicious accounts at all. One interesting finding is that the Stringhini approach indeed performs better than the two

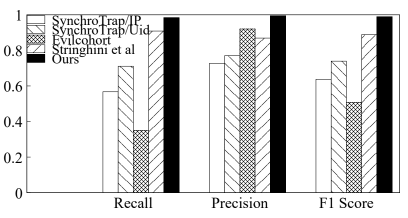


Figure 7: Comparison with baselines

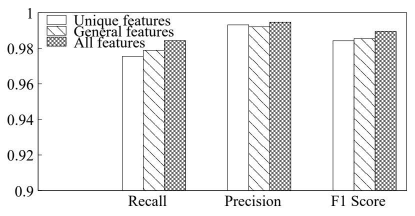


Figure 8: General and/or WLink unique features.

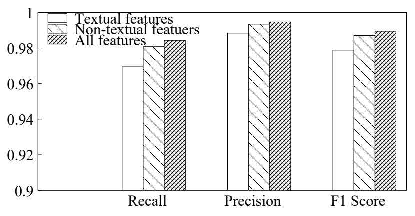


Figure 9: Textual and/or non-textual features.

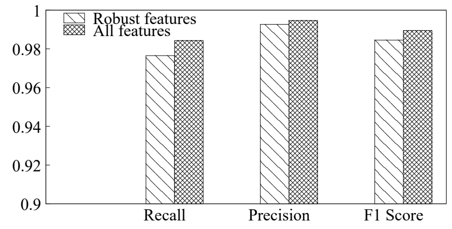


Figure 10: Robust and/or non-robust features.

state-of-the-art approaches based on community detection. This confirms our hypothesis that the adversaries on WLink are not sophisticated enough to bypass simpler detection approaches, and also they tend to not form a community. However, all these approaches' performance is significantly lower than Realguard. Note that we have also trained our model on an older dataset and tested it on a newer dataset, and the result is similar. The aucs of each approach are also computed and the results are similar. They are not included due to the space limitation.

In particular, we can observe that our approach significantly improves over previous approaches to achieve a precision of 99.5%, a recall of 98.4%, and an F1 score of 98.9%, while prior approaches cannot achieve an F1 score higher than 88.8%. We attribute this to the synergy of two effects: (1) WLink employs a privacy-centric design to move the battleground between the attacker and the detector to friend requests; (2) our Realguard approach is specifically designed to detect malicious accounts based on friend requests. In the following, we will present more ablation study results to provide further insights into different classes of features.

5.2.2 Ablation study. We now conduct various experiments to understand the effect of different categories of features. Figure 8 shows the comparison of generic features versus WLink unique features. We observe that unique features can achieve a high recall (i.e., above 97%); however, we also observe that using only WLink unique features is not as effective as using only general features. We attribute it to the fact that WLink unique features are a small subset of all features. In addition, we observe that using only general features cannot achieve the highest recall; this shows that WLink unique features are indeed helpful to detect malicious WLink accounts.

In terms of precision, in contrast, we observe that the model can achieve a high precision using any set of the features; and using only WLink unique features' precision is higher than using only general features.

Understanding textual features. In Figure 9, we present the comparison results of (1) using only textual features; (2) using only non-textual features; and (3) using all features. We observe that using textual features only can achieve a recall over 96% and precision over 98%, showing that the request messages capture essential information to detect malicious accounts effectively and accurately.

Also, we observe that using both textual and non-textual features improves over using non-textual features only in terms of both precision and recall with a small margin. Therefore, we conclude that textual features can help to improve the model's performance, but the improvement is limited.

Using only robust features achieves comparable performance to using all features. We evaluate whether using only robust features can yield reasonably good performance. The evaluation results are presented in Figure 10. We observe that using only robust features will decrease the performance, but the margin is as small as 1%. The most significant drop in performance is on recall, which drops from 98% to 97%. However, both of these meet the standard for real deployment. Therefore, we conclude that using robust features do not hurt the performance by much.

On the other hand, by its definition, using only robust features will make adversaries harder to evade the detector. Therefore, when evasion attacks are in consideration, Realguard provides a robust and effective solution against such attackers.

5.2.3 Understanding the importance of each feature. We try to understand the importance of each feature from the finally trained random forest. They are plotted in Figure 11. We observe that among the top-10 most important features, 6 of them are derived from friend requests, while only 4 are account-related features; among the top-6 friend request-related features, the most important two features are textual features. These confirm our hypothesis that friend requests, especially request messages, capture more information to distinguish malicious accounts from benign ones. We also highlight the robust features in the red color. We can observe that robust features play important roles in the decision trees. This explains why using only robust features can also achieve high performance.

5.3 Real World Deployment

We deploy our Realguard prototype in WLink's production pipeline to handle their daily dataset. This unlabeled dataset consists of more than one billion friend requests sent from 20 million accounts with a time span of one day. As explained in Section 5.1, we can tune the threshold on the Random Forest output for better performance.

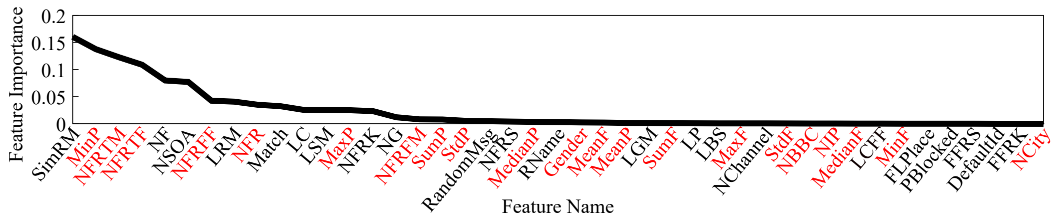


Figure 11: Feature importance. Robust features are highlighted in red. Feature explanations can be found in the appendix.

In this experiment, we set it to be 0.9 to prefer better precision (i.e., reducing the false positive rate).

Realguard can detect 0.59M malicious accounts every day. In comparison, the previous deployment of the automatic malicious account detection system can only detect 0.1M ones; thus Realguard can detect 6× more malicious accounts than the previous deployment. We randomly sample some accounts predicted as malicious and send them to WLink’s security department for manual inspection. It is reported that 90% of the sampled malicious accounts are indeed malicious. This precision meets the deployment standard of WLink, and is also comparable to the previous deployment.

We analyze those false positives and find out that a large proportion of them are those sending too many friend requests through location-based services and containing advertisement in request messages, which look indeed malicious without further information. We conclude that our approach is effective at detecting malicious accounts in the real mobile social network WLink.

6 RELATED WORK

Most malicious accounts detection methods can be roughly divided into two categories, i.e., *feature-based methods* and *social-structure-based methods*. Our method is a feature-based one.

6.1 Feature-based Methods

These methods [5, 12, 15–17, 19, 23, 27, 42, 43, 46, 48, 54] represent accounts using various features extracted from their profiles, social graphs, behaviors, content, and/or interactions. Some example features include the number of friends/followers, the ratio between the number of followees and the number of followers, clustering coefficients, number of URLs in tweets, the similarity between the messages/tweets sent by an account, hashtags in tweets, and/or clickstream patterns.

Then, they leverage machine learning techniques to process these features to identify malicious accounts. Specifically, some methods [5, 17, 23, 27, 42, 43, 48, 54] use supervised machine learning techniques to learn classifiers from a training dataset consisting of both labeled benign accounts and labeled malicious accounts, and then the classifiers are used to classify the remaining accounts to be benign or malicious. Some other methods [12, 23] leverage clustering techniques to group accounts based on their feature similarities; and accounts that form clusters are more likely to be predicted as malicious. For instance, SynchroTrap [12] identified malicious accounts with synchronized behaviors via hierarchical clustering and was deployed by Facebook.

However, these studies focused on web OSNs. Malicious accounts in web OSNs and mobile OSNs may have different characteristics due to the intrinsic differences in design and functionality between web and mobile OSNs. Therefore, the features that are effective for detecting malicious accounts in web OSNs may not be effective for mobile OSNs. Moreover, these studies did not consider the unique features of mobile OSNs.

6.2 Social-Structure-based Methods

These methods model an OSN as a graph, where nodes are accounts and edges represent social relationships (e.g., friendship, following, interaction) between accounts. They aim to identify the structural anomaly between benign accounts and malicious accounts in such a social graph. Specifically, to analyze the structure of the social graph, they leverage random walks [11, 14, 24, 34, 41, 53, 55, 56], community detection [9, 20, 47], or Loopy Belief Propagation (LBP) [29, 30, 49–52].

Among the social-structure-based methods, LBP-based methods were shown to outperform random walk based and community detection based methods [29, 49–52]. Specifically, LBP-based methods model a social graph as pairwise Markov Random Fields and leverage LBP or its variants to perform inference. Roughly speaking, each node in the social graph is associated with a binary random variable. A pairwise Markov Random Field models the joint probability distribution of all the binary random variables, where the statistical dependence/correlations are captured by the graph structure. Given a set of labeled benign accounts and/or labeled malicious accounts, LBP-based methods leverage LBP [29, 30] or linearized LBP [49, 51, 52] to infer the conditional probability distribution for each binary random variable, which is used to predict a node to be benign or malicious. Linearized LBP guarantees convergence, while the standard LBP does not. Moreover, linearized LBP is much more efficient than the standard LBP. Most LBP-based methods assign a constant weight to edges in the graph. Wang et al. [50] recently proposed a method to learn the edge weights.

Social-structure-based methods can be combined with feature-based methods. Specifically, we can first use a feature-based method to predict the probability of being malicious for each account. Such a probability of being malicious can be treated as prior knowledge and incorporated into an LBP-based method. Indeed, Gao et al. [29] showed that combining feature-based methods and LBP-based methods can further enhance detection accuracy.

7 CONCLUSION

In this paper, we present the first work on detecting malicious PC-MSN accounts. We conducted a large-scale measurement study over a collection of 1 million labeled data from WLink, a real PC-MSN with billions of users, and the results show that friend requests are useful indicators for malicious account detection. We proposed a set of new features, and analyze their robustness against evasion attacks. We then propose a supervised learning approach to combine deep neural networks and random forest for final prediction. We implemented a prototype, Realguard, and our evaluation shows that Realguard can outperform previous state-of-the-art approaches in terms of both precision and recall. Also, we show that using only robust features can achieve the same level of performance as using all features. We deploy Realguard in WLink's production pipeline, and Realguard can process WLink's daily transactions in 2 hours while detecting 0.59 million accounts on average each day, which is 6× of the performance of the previous deployed approaches. A manual verification shows that Realguard's precision in predicting malicious accounts can achieve over 90%. These results show that Realguard is effective and accuracy at detecting malicious PC-MSN accounts in WLink.

8 ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 61572278, U1736209, and 61872211. Qi Li is the corresponding author of this paper.

REFERENCES

[1] 2018. Jaccard index. https://en.wikipedia.org/wiki/Jaccard_index.
[2] 2018. KakaoTalk. https://www.kakaocorp.com/service/KakaoTalk.
[3] 2018. LINE. https://line.me/en-US/.
[4] 2018. Renren. http://renren.com/.
[5] Fabrıcio Benevenuto, Gabriel Magno, Tiago Rodrigues, and Virgılio Almeida. 2010. Detecting spammers on twitter. In *CEAS*.
[6] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda. 2009. All Your Contacts Are Belong to Us: Automated Identity Theft Attacks on Social Networks. In *WWW*.
[7] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *JSTAT* 2008, 10 (2008), P10008.
[8] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32. https://doi.org/10.1023/A:1010933404324
[9] Zhuhua Cai and Christopher Jermaine. 2012. The Latent Community Model for Detecting Sybils in Social Networks. In *NDSS*.
[10] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Kamesh Munagala. 2015. Combating friend spam using social rejections. In *ICDCS*.
[11] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. 2012. Aiding the Detection of Fake Accounts in Large Scale Social Online Services. In *NSDI*.
[12] Qiang Cao, Xiaowei Yang, Jieqi Yu, and Christopher Palow. 2014. Uncovering Large Groups of Active Malicious Accounts in Online Social Networks. In *CCS*.
[13] Zi Chu, Indra Widjaja, and Haining Wang. 2012. Detecting Social Spam Campaigns on Twitter. In *ACNS*.
[14] G. Danezis and P. Mittal. 2009. SybilInfer: Detecting Sybil Nodes using Social Networks. In *NDSS*.
[15] Manuel Egele, Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. 2013. Compa: Detecting compromised accounts on social networks.. In *NDSS*.
[16] Bimal Viswanath et al. 2014. Towards Detecting Anomalous User Behavior in Online Social Networks.. In *USENIX Security Symposium*.
[17] Chao Yang et al. 2011. Die Free or Live Hard? Empirical Evaluation and New Design for Fighting Evolving Twitter Spammers. In *RAID*.
[18] Gang Wang et al. 2013. Social Turing Tests: Crowdsourcing Sybil Detection. In *NDSS*.
[19] Haizhong Zheng et al. 2018. Smoke Screener or Straight Shooter: Detecting Elite Sybil Attacks in User-Review Social Networks. In *NDSS*.
[20] Lorenzo Alvisi et al. 2013. SoK: The Evolution of Sybil Defense via Social Networks. In *IEEE S & P*.
[21] Saptarshi Ghosh et al. 2012. Understanding and Combating Link Farming in the Twitter Social Network. In *WWW*.
[22] Shirin Nilizadeh et al. 2017. POISED: Spotting Twitter Spam Off the Beaten Paths. In *CCS*.
[23] Wang Gang et al. 2013. You Are How You Click: Clickstream Analysis for Sybil Detection.. In *USENIX Security Symposium*.
[24] Yazan Boshmaf et al. 2015. Integro: Leveraging Victim Prediction for Robust Fake Account Detection in OSNs.. In *NDSS*.
[25] Z. Yang et al. 2016. VoteTrust: Leveraging Friend Invitation Graph to Defend against Social Network Sybils. *IEEE TDSC* 13, 4 (2016), 488–501.
[26] Emilio Ferrara, Onur Varol, Clayton A. Davis, Filippo Menczer, and Alessandro Flammini. 2016. The rise of social bots. *Commun. ACM* 59 (2016), 96–104.
[27] Hongyu Gao, Yan Chen, Kathy Lee, Diana Palsetia, and Alok Choudhary. 2012. Towards online spam filtering in social networks. In *NDSS*.
[28] Hongyu Gao, Jun Hu, Christo Wilson, Zhichun Li, Yan Chen, and Ben Y Zhao. 2010. Detecting and characterizing social spam campaigns. In *IMC*.
[29] Peng Gao, Binghui Wang, Neil Zhenqiang Gong, Sanjeev R Kulkarni, Kurt Thomas, and Prateek Mittal. 2018. Sybilfuse: Combining local attributes with global structure to perform robust sybil detection. In *CNS*.
[30] Neil Zhenqiang Gong, Mario Frank, and Prateek Mittal. 2014. SybilBelief: A Semi-supervised Learning Approach for Structure-based Sybil Detection. *IEEE TIFS* 9, 6 (2014).
[31] Chris Grier, Kurt Thomas, Vern Paxson, and Michael Zhang. 2010. @spam: The Underground on 140 Characters or Less. In *CCS*.
[32] Hacking Election. 2016. http://goo.gl/G8o9x0
[33] Hacking Financial Market. 2016. http://goo.gl/4AkWyt
[34] Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. 2017. Random Walk based Fake Account Detection in Online Social Networks. In *IEEE DSN*.
[35] Yoon Kyung Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*.
[36] Kyumin Lee, James Caverlee, and Steve Webb. 2010. Uncovering social spammers: social honeypots+ machine learning. In *SIGIR*.
[37] Kyumin Lee, Brian David Eoff, and James Caverlee. 2011. Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter.. In *ICWSM*.
[38] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent Neural Network for Text Classification with Multi-Task Learning. In *IJCAI*.
[39] Juan Martinez-Romo and Lourdes Araujo. 2013. Detecting malicious tweets in trending topics using a statistical analysis of language. *Expert Syst. Appl.* 40, 8 (2013), 2992–3000.
[40] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR Workshop*.
[41] Abedelaziz Mohaisen, Nicholas Hopper, and Yongdae Kim. 2011. Keep your friends close: Incorporating trust into social network-based Sybil defenses. In *INFOCOM*.
[42] Jonghyuk Song, Sangho Lee, and Jong Kim. 2011. Spam filtering in Twitter using sender-receiver relationship. In *RAID*.
[43] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. 2010. Detecting spammers on social networks. In *ACSAC*.
[44] Gianluca Stringhini, Pierre Mourlanne, Gregoire Jacob, Manuel Egele, Christopher Kruegel, and Giovanni Vigna. 2015. Evilcohort: detecting communities of malicious accounts on online services. In *USENIX Security Symposium*.
[45] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. 2011. Design and evaluation of a real-time url spam filtering service. In *IEEE S & P*.
[46] Kurt Thomas, Chris Grier, Vern Paxson, and Dawn Song. 2011. Suspended Accounts in Retrospect: An Analysis of Twitter Spam. In *IMC*.
[47] Bimal Viswanath, Ansley Post, Krishna P. Gummadi, and Alan Mislove. 2010. An Analysis of Social Network-Based Sybil Defenses. In *SIGCOMM*.
[48] Alex Hai Wang. 2010. Don't Follow Me - Spam Detection in Twitter. In *SECRYPT 2010*.
[49] Binghui Wang, Neil Zhenqiang Gong, and Hao Fu. 2017. GANG: Detecting Fraudulent Users in Online Social Networks via Guilt-by-Association on Directed Graphs. In *ICDM*.
[50] Binghui Wang, Jinyuan Jia, and Neil Zhenqiang Gong. 2019. Graph-based security and privacy analytics via collective classification with joint weight learning and propagation. (2019).
[51] Binghui Wang, Jinyuan Jia, Le Zhang, and Neil Zhenqiang Gong. 2018. Structure-based sybil detection in social networks via local rule-based propagation. *IEEE Transactions on Network Science and Engineering* (2018).
[52] Binghui Wang, Le Zhang, and Neil Zhenqiang Gong. 2017. SybilSCAR: Sybil Detection in Online Social Networks via Local Rule based Propagation. In *INFOCOM*.
[53] Chao Yang, Robert Harkreader, Jialong Zhang, Seungwon Shin, and Guofei Gu. 2012. Analyzing Spammer's Social Networks for Fun and Profit. In *WWW*.
[54] Zhi Yang, Christo Wilson, Xiao Wang, Tingting Gao, Ben Y. Zhao, and Yafei Dai. 2011. Uncovering Social Network Sybils in the Wild. In *IMC*.
[55] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. 2008. SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks. In *IEEE S & P*.
[56] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. 2006. SybilGuard: Defending Against Sybil Attacks via Social Networks. In *SIGCOMM*.

A REPRODUCIBILITY MATERIALS

A.1 Full feature set

The features that we use are summarized in Table 2. The design of these features is motivated by our measurement study. Most of the features are self-explained. We classify all features to be textual and non-textual.

The textual features include (1) the number and the fraction of the request messages include "I am" (the text from the template); (2) the similarity between request messages; and (3) six numeric features emitted by a DNN-based module. The first two have been explained in Section 3.3. We will explain the details of the DNN model later. We note that most of the features we use have not been proposed before, and some of them are unique to mobile social networks. In Section 5, we will evaluate our approach, and show the importance of these features.

Feature robustness. As we mentioned above, we consider a feature to be robust, if (1) the attacker cannot manipulate the value of the feature; (2) the attacker needs to incur significant cost to do so; or (3) the attacker's utility will be degraded in doing so. We now discuss the features that we think are robust, and explain why.

The only robust account-related features are gender and the number of bound bank cards. Since banks require verifying the real identity of their customers to open an account and to get a card, it will incur a significant cost for a malicious account to bind many bank cards, and will also increase the chance to unveil the real identity of the attacker. Therefore, we think this feature is robust.

Most of the robust friend requests-related features have been discussed in Section 3. NIP and NCity are related to the IP addresses and their cities. Like in traditional online social networks, attackers need to maintain multiple IP addresses to login to their malicious accounts, so that they can still get access to their accounts if one IP address is banned by WLink due to malicious activities. Typically benign users do not have such a requirement. Therefore, we consider NIP and NCity as robust features. For a friend request, whether the sender and the receiver has a common friend is an indicative factor for two users to know each other in person, and it is hard for a malicious account to forge. Since WLink does not allow users to see the friends of his friend, even though an attacker may be able to become a friend of a benign user (victim), it is not easy for him to leverage this property to send friend requests to the victim's friends. Therefore, we consider the six related features (SumF, MeanF, MaxF, MinF, StdF, MedianF) as robust.

DNN-based feature extractor. Now we detail the DNN-based feature extractor. Given a request message m, we want to learn a function f, so that $f(m) \in [0, 1]$ indicates the probability of m being a good request message.

Thanks to the effectiveness of deep learning, we can model f as a deep neural network. Given m is a sequence of characters with a variable length, we can model f as either a *recurrent neural network* (RNN) [38], or a *convolutional neural network* (CNN) [35]. In evaluation, we observe that the CNN approach deliver better runtime performance, and thus we choose to use a CNN to model f. The concrete neural network architecture is provided in Figure 6. In particular, in the second step, we use the same word2vec model to convert a message into a matrix as described in Section 3.3. In the end, the softmax output is a 2-dimensional vector, where its first dimension's value is outputted as $f(m)$.

During prediction, once the message m is mapped into $f(m)$, SumP is computed as the summation of $f(m)$ for all messages.The other five aggregated metrics are computed similarly.

A.2 Evaluation details

Implementation details. We implement our prototype, namely Realguard, using both Scala and Python. In particular, we implement the word vector training using the word2vec implementation from Spark MLlib[2], which uses skip-gram model. The rest is implemented in Python. The neural network component to extract features from request messages is implemented in Tensorflow[3]. All other components for the feature extraction are implemented using the PySpark[4]. The tokenizer is implemented using Jieba[5]. We use the Random Forest implementation in Spark MLlib for the final classifier.

For training, the neural network handling natural language is trained with early stopping, and it stops when the loss gets higher on the validation set. To achieve higher precision and recall, we tried different combinations of hyperparameters and chose the final hyperparameters according to the model's performance on the validation set. Then we test our models on the test set.

For testing, Random Forest can produce a prediction score ranging from 0 to 1, and larger scorer indicates a higher likelihood of an instance being malicious. We can set a threshold to classify each instance as malicious or benign. In all our evaluation, we set the threshold to be 0.5. However, we can adjust this threshold to trade off between precision and recall.

Baselines. We compare our methods with the state-of-the-art solutions, SynchroTrap [12], Evilcohort [44], and Stringhini et al. [43]. The first two are the state-of-the-art community-detection-based approaches. However, one may wonder whether the adversary on WLink has not been too sophisticated, so maybe earlier detection algorithm is still effective. Therefore, we chose Stringhini [43] as such a representative, which fits the WLink scenario the best. We perform hyper-parameter tuning of these approaches, choose the one for each of them to achieve the best performance. We briefly explain these methods below.

- **SyncroTrap/uid** builds a graph of accounts as nodes with each edge associated with a weight to be the *similarity* between the two nodes connected computed using Jaccard index [1] according to uid of the request receiver. Only edges with a weight greater than a threshold will be preserved, and a community is defined as a connected sub-graph. Large communities will be considered malicious. Similarly, **SyncroTrap/IP** builds a graph of accounts as nodes and creates the graph similarly according to the IP addresses associated with the friend requests.
- **Evilcohort** uses the bipartite graph between accounts and IP addresses to detect communities using the Louvain modularity [7] and consider communities as malicious.

[2]https://spark.apache.org/mllib/
[3]https://tensorflow.org
[4]http://spark.apache.org/docs/latest/api/python/
[5]https://github.com/fxsjy/jieba

Category		Feature Name (Acronym)	Proposed by us	WLink unique	Robust
Account -related	Non-textual features	Gender (Gender)	[24]		✓
		Whether default WLink id is used (DefaultId)	✓	✓	
		Whether it has frequently logined place (FLPlace)	✓		
		Whether its posts has been blocked (PBlocked)	✓	✓	
		Whether it has real name (RName)	✓	✓	
		Whether it used RandomMsg (RandomMsg)	✓	✓	
		Whether it used LBS (LBS)	✓	✓	
		Whether it used Match (Match)	✓	✓	
		Number of bound bank cards (NBBC)	✓	✓	✓
		Number of groups (NG)	✓		
		Number of friends (NF)	[43]		
		Number of subscribed official accounts (NSOA)	✓	✓	
		Liveness of sending messages (LSM)	[27]		
		Liveness of receiving messages (LRM)	✓		
		Liveness of receiving group messages (LGM)	✓		
		Liveness of posting (LP)	[37]		
		Liveness of comments (LC)	[26]		
		Liveness of comments from friends (LCFF)	[39]		
Friend requests -related		Number of friend requests (NFR)	[54]		✓
		Number of friend requests in sleeping time (NFRS)	✓		
		Fraction of friend requests in sleeping time (FFRS)	✓		
		Number of friend requests sent from male (NFRFM)	✓		✓
		Number of friend requests sent from female (NFRFF)	✓		✓
		Number of friend requests sent to male (NFRTM)	✓		✓
		Number of friend requests sent to female (NFRTF)	✓		✓
		Number of unique IP addresses (NIP)	✓		✓
		Number of unique cities based on IP addresses (NCity)	✓		✓
		Number of unique channels (NChannel)	✓	✓	
		Sum, mean,max,min,standard deviation and median of attribute whether the sender and the receiver have common friends (SumF, MeanF, MaxF, MinF, StdF, MedianF)	✓		✓
	Textual features	Number of request messages containing "I am" (NFRK)	✓	✓	
		Fraction of request messages containing "I am" (FFRK)	✓	✓	
		Sum, mean, max, min, std and median of probabilities of request message being malicious (SumP, MeanP, MaxP, MinP, StdP, MedianP)	✓	✓	✓
		Similarity of request messages (SimRM)	✓	✓	

Table 2: The features used in our study.

Sender	The sender’s ID
Receiver	The receiver’s ID
Timestamp	The timestamp when the friend request is sent
IP	The sender’s IP address
Channel	The channel uses to send the friend request
Message	The request message along with the request

Table 3: Fields in a friend request.

	Name	Description
1	Group	Via group members
2	ID	Via WLink ID
3	Number	Via phone number
4	NameCard	Via name card
5	QRCode	Via scanning QR code
6	LBS	Via location-based service
7	RandomMsg	Via random message service
8	Match	Via match service

Table 4: Channels of sending friend requests in WLink**.**

- **Stringhini et al.** extracts features including *Friend Number* (number of friends), *Messages Sent* (number of messages sent), *FF ratio* (number of friend requests that a user sent to the number of friends she has), *Friend Choice* (the total number of names among the profiles’ friend to the number of distinct first names), *Message Similarity* (the similarity among the messages sent by a user), *URL ratio* (number of messages containing urls to the number of messages). They then use Random Forest as their classifier.