Detecting Fake Accounts in Online Social Networks at the Time of Registrations

Dong Yuan[1,2], Yuanli Miao[1,2], Neil Zhenqiang Gong[3], Zheng Yang[1,2], Qi Li[1,2], Dawn Song[4], Qian Wang[5], Xiao Liang[6]

[1]Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China
[2]Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing, China
[3]Department of Electrical and Computer Engineering, Duke University, Durhamn, NC
[4]Computer Science Division, University of California, Berkeley, CA
[5]School of Cyber Science and Engineering, Wuhan University, China
[6]Tencent, China

{yuand15@mails, myl16@mails, yz17@mails, qli01@}tsinghua.edu.cn, neil.gong@duke.edu, dawnsong@cs.berkeley.edu, qianwang@whu.edu.cn, doodleliang@tencent.com

ABSTRACT

Online social networks are plagued by fake information. In particular, using massive fake accounts (also called Sybils), an attacker can disrupt the security and privacy of benign users by spreading spam, malware, and disinformation. Existing Sybil detection methods rely on rich content, behavior, and/or social graphs generated by Sybils. The key limitation of these methods is that they incur significant delays in catching Sybils, i.e., Sybils may have already performed many malicious activities when being detected.

In this work, we propose Ianus, a Sybil detection method that leverages account registration information. Ianus aims to catch Sybils immediately after they are registered. First, using a real-world registration dataset with labeled Sybils from WeChat (the largest online social network in China), we perform a measurement study to characterize the registration patterns of Sybils and benign users. We find that Sybils tend to have *synchronized* and *abnormal* registration patterns. Second, based on our measurement results, we model Sybil detection as a graph inference problem, which allows us to integrate heterogeneous features. In particular, we extract synchronization and anomaly based features for each pair of accounts, use the features to build a graph in which Sybils are densely connected with each other while a benign user is isolated or sparsely connected with other benign users and Sybils, and finally detect Sybils via analyzing the structure of the graph. We evaluate Ianus using real-world registration datasets of WeChat. Moreover, WeChat has deployed Ianus on a daily basis, i.e., WeChat uses Ianus to analyze newly registered accounts on each day and detect Sybils. Via manual verification by the WeChat security team, we find that Ianus can detect around 400K per million new registered accounts each day and achieve a precision of over 96% on average.

CCS '19, November 11–15, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6747-9/19/11…$15.00
https://doi.org/10.1145/3319535.3363198

CCS CONCEPTS

• **Security and privacy → Security services**; **Network security**.

KEYWORDS

Fake account detection; Sybil detection

ACM Reference Format:
Dong Yuan, Yuanli Miao, Neil Zhenqiang Gong, Zheng Yang, Qi Li, Dawn Song, Qian Wang, Xiao Liang. 2019. Detecting Fake Accounts in Online Social Networks at the Time of Registrations. In *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS'19), November 11-15, 2019, London, United Kingdom.* ACM, NewYork, NY, USA, 16 pages. https://doi.org/10.1145/3319535.3363198

1 INTRODUCTION

Online social networks, e.g., Facebook and WeChat, are particularly popular nowadays, which have a huge influence on our daily life. Meanwhile, they are also important targets for attackers. For instance, in a Sybil attack [14], an attacker registers and maintains massive fake accounts to perform various malicious activities such as spreading spams, phishing URLs, malware, and disinformation [21, 35] as well as stealing private user data [4].

Many methods [2, 6, 10–13, 16–18, 20, 27–30, 33, 34, 37–40, 44, 49–51, 54, 55, 57, 58] have been developed to detect Sybils in online social networks. These methods leverage content (e.g., URLs in tweets), behavior (e.g., clickstreams, likes, photo uploads), and/or social graphs (e.g., friendship graphs, follower-followee graphs) generated by Sybils. They face a key limitation: they incur significant delays at detecting Sybils. Specifically, they require Sybils to generate rich content, behavior, and/or social graphs before detecting them. Therefore, Sybils may have already performed various malicious activities before being detected.

In this work, we aim to detect Sybils at the time of registration. CAPTCHA seems a natural choice to prevent registrations of Sybil accounts. However, a number of studies (e.g., [7–9, 52]) have shown that an attacker can use machine learning techniques to automatically bypass CAPTCHAs, making them ineffective. We propose to detect Sybils based on registration information.

First, using an anonymized real-world registration dataset from WeChat, an online social network designed for mobile devices, we perform systematic measurement studies about the registration patterns of Sybils and benign users. The dataset is collected in November, 2017, 770K of which are benign users and 647K of which are Sybils. Each registration has a list of *attributes* such as IP address, phone number (can be used as a user ID in WeChat), device ID (e.g., IMEI), nickname, etc..[1] We find that Sybils share *synchronized* registration patterns with respect to these attributes. For instance, many Sybils use IP addresses with the same 24-bit IP prefixes for registration, which means that they could be registered in clusters from the same subnetwork. However, synchronization is insufficient to distinguish between Sybils and benign users. In particular, some benign users also share synchronized registration attributes, e.g., they use IPs with the same prefix. Whether two accounts that share synchronized registration attributes are Sybils or not further depends on the risk/anomaly of the synchronized attributes. Therefore, we further study *abnormal* registration patterns. For instance, when two Sybils use the same IP prefix for registration, the IP prefix itself is abnormal, e.g., many registrations from the IP prefix are during late night. However, when two benign users use the same IP prefix for registration, the IP prefix is less abnormal, e.g., accounts from the IP prefix are mainly registered in daytime.

Second, based on our measurement results, we design Ianus to detect Sybils using their registration data. A key challenge is how to integrate the synchronized and abnormal registration patterns as they are heterogeneous. Ianus leverages graph inference techniques to address the challenge. Specifically, our goal is to leverage the synchronized and abnormal registration patterns to build a weighted graph to represent the complex relationships between accounts, and then we detect Sybils via analyzing the structure of the graph. We build the graph in a way such that each node is an account, Sybils are densely connected with each other, while benign accounts are sparsely connected with each other and with Sybils. To achieve the goal, Ianus has three components, i.e., *feature extraction*, *graph building*, and *Sybil detection*. In feature extraction, for a pair of accounts, we extract binary *synchronization based features* (e.g., the two accounts use the same device for registration) and *anomaly based features* (e.g., the shared device is abnormal), which are inspired by our measurements on the synchronized and abnormal registration patterns, respectively.

In graph building, we construct a weighted graph between accounts using their synchronization based features and anomaly based features. Specifically, we assign a *sync-anomaly* score for a pair of accounts based on their features. A sync-anomaly score characterizes the synchronization and anomaly patterns between two accounts, and a pair of accounts have a higher sync-anomaly score if they have more synchronized and abnormal registration attributes. One way to assign a sync-anomaly score for a pair of accounts is to sum their binary features and treat the sum as the sync-anomaly score. However, such feature-sum-based method does not consider features' different weights. Therefore, we use machine learning techniques (logistic regression in particular) to learn *sync-anomaly* scores that automatically weight different features. Then, we create edges between accounts using their sync-anomaly scores. In particular, to build a graph in which only Sybils are densely connected with each other, we create an edge between two accounts only if both of them are predicted as Sybils based on their sync-anomaly score. Moreover, we treat the sync-anomaly score as the weight of the edge.

In the Sybil detection component, community detection is a natural choice to detect Sybils as Sybils are densely connected in our graph. Moreover, we design a simple weighted node degree based method to detect Sybils. Specifically, a node in our graph has a larger weighted degree if it is connected to more Sybils by edges with larger weights, which indicates that the node is more likely to be Sybil. Therefore, Ianus can predict a node to be a Sybil if its weighted degree is large enough.

Third, we evaluate Ianus using registration datasets from WeChat. Ianus can detect a large fraction of Sybils (recall is 80.2%) with a high precision (92.4%). Moreover, we extensively evaluate different design choices for each component of Ianus. Specifically, in feature extraction, our results show that the synchronization based features and anomaly based features are complementary and combining them does improve detection accuracy. In graph building, we find that learning the sync-anomaly scores using logistic regression outperforms the feature-sum-based sync-anomaly scores. In the Sybil detection component, we evaluate the popular Louvain method [5] as a community detection method. Our results show that the Louvain method and our weighted node degree based method achieve close detection accuracies. However, our weighted node degree based method is more efficient than the Louvain method. For instance, the Louvain method takes 40 minutes to analyze the graph, while our weighted node degree based method takes only 10 minutes. WeChat has deployed Ianus on a daily basis, i.e., applying Ianus to detect Sybils among the accounts registered within each day. The security team of WeChat confirmed that Ianus can detect around 400K per million new registered accounts per day with a precision of over 96% on average.

In summary, we have the following contributions:

- We perform a large-scale measurement study to characterize the registration patterns of Sybils and benign users. We find that Sybils have both synchronized and abnormal registration patterns.
- We propose Ianus to detect Sybils using registration data. Ianus builds a weighted graph for accounts based on their synchronized and abnormal patterns, and then detects Sybils via analyzing the graph's structure.
- We evaluate Ianus using registration datasets from WeChat. Moreover, WeChat has deployed Ianus, demonstrating the industrial impact of Ianus.

2 RELATED WORK

Existing Sybil detection methods [2, 3, 6, 10–13, 16–18, 20, 27–30, 33, 34, 37–40, 44, 47, 49–51, 54–58] for online social networks mainly leverage content, behavior, and/or social graphs generated by Sybils. Content and behavior based methods [12, 16, 18, 30, 33, 38, 40, 44, 49] often model Sybil detection as a binary classification problem and leverage machine learning techniques. Specifically, they first extract features from a user's content (e.g., posts,

[1] We performed experiments at WeChat who anonymized the dataset to the extent such that the dataset is just enough for our study.

tweets) and/or behavior (e.g., frequency of tweeting, likes, photo uploads, and clickstreams). Then, they detect Sybils via analyzing the features through either supervised machine learning techniques (e.g., logistic regression) or unsupervised machine learning techniques (e.g., community detection, clustering). For instance, SynchroTrap [12] leverages clustering techniques to detect Sybils based on synchronized user behavior, e.g., photo uploading.

EvilCohort [34] leverages users' IP addresses to detect Sybils. Specifically, given users' logins, EvilCohort builds a user-IP bipartite graph, where an edge between a user and an IP means that the user once logged in from the IP. Then, EvilCohort uses the Louvain method to detect Sybils in the user-IP graph. EvilCohort may require multiple user logins such that Sybils form communities in the user-IP bipartite graph. During the multiple logins, Sybils may have already performed malicious activities. Our method does not require user logins except the registrations.

Social graph based methods [2, 6, 10, 11, 13, 17, 20, 27, 28, 37, 39, 50, 51, 54, 55, 57, 58] detect Sybils via leveraging graph-based machine learning techniques (e.g., random walks [6, 11, 13, 23, 28, 50, 51, 54, 55], belief propagation [19, 20, 41–43], and community detection [10, 39]) to analyze the structure of the social graphs between users. The key limitation of these methods is that they incur significant delays at detecting Sybils because they rely on rich content, behavior, and/or social graphs generated by Sybils. Ianus addresses this limitation by leveraging registration data to detect Sybils.

Hao et al. [22] proposed PREDATOR to detect malicious domains at time of registration. One key difference is that detecting malicious domains and detecting Sybils require different features, e.g., our Ianus leverages features from the mobile device, OS, user location, and nickname. Xie et al. [48] proposed Souche for early recognition of legitimate users. Different from Ianus, Souche relies on vouching activities between users, e.g., users send messages to each other. Leontjeva et al. [26] proposed a method to detect Sybils in Skype. Instead of using only registration features, they leverage a variety of features, which require the accounts to have generated enough content and social data.

The work by Thomas et al. [38] is perhaps the most related to ours. They leveraged registration data to detect Sybils in Twitter. However, they leveraged the registration data (e.g., signup flow, user agents, form submission timing) that are specific to Twitter, while Ianus leverages different registration data such as IPs, phone numbers, devices, etc..

3 MEASURING REGISTRATION PATTERNS

We briefly review WeChat, a popular online social network in China, and use a real WeChat dataset to measure the registration patterns of Sybils and benign users. These measurements will be the basis of our Ianus.

3.1 WeChat and Dataset

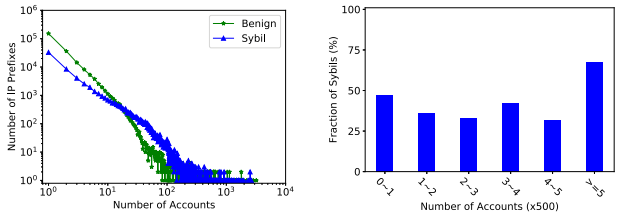
WeChat is the largest mobile social network application in China with over 1 billion monthly active users. Users can only register accounts on WeChat through mobile devices. With an account, a user can send instant text, voice, and video messages to their friends via group chat or one-to-one chat, as well as publish posts. We obtained a user registration dataset from WeChat. The dataset was collected in November, 2017. Specifically, the dataset includes 1.4M registered accounts, consisting of roughly 0.77M benign accounts and 0.65M Sybil accounts. The labels were obtained from WeChat's existing behavior-based Sybil detection system (this system predicts the labels months after these accounts were registered as it requires sufficient account behavior) and user reports (users can report others as Sybil in WeChat). The security team of WeChat manually inspected some randomly sampled accounts and found that the accuracy is more than 95%. We acknowledge that 95% accuracy instead of 100% may introduce bias to our measurement results. However, we believe such bias should be small and our measurement results are representative.

Table 1: Registration attributes.

Attribute	Example
IP	**********
Phone Number	+86-157-7944-xxxx
Timestamp	1499270558
Nickname	***
WeChat Version	6.6.7
OS Version	iOS 10.3.2
Hashed WiFi MAC	a9d0cf034aa4e113e8ca27e9110928c7
Hashed Device ID	d5c027d91d1df579d6ad1bffbb638cee

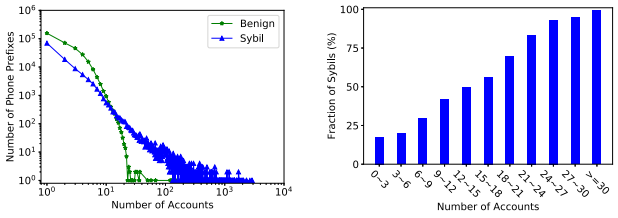
Registration attributes: Each registration has a list of attributes, which are shown in Table 1. The Phone Number is the number a user used to register an account. Each number can only register one account and can be used as account ID. In China, a phone number is in the format +86-xxx-xxxx-xxxx, where the first two digits +86 is the country code of China, the next three digits represent a mobile phone service provider (e.g., China Unicom), the middle four digits represent an area code (e.g., mobile phone numbers obtained from Beijing have specific area codes), and the last four digits indicate a customer code. A user can also specify a Nickname, which could be a combination of digits, English characters, Chinese characters, and other special characters. The WeChat Version (e.g., 6.6.7) and OS Version (e.g., iOS 10.3.2) indicate the WeChat app and mobile operating system that were used to register an account, respectively. WiFi MAC is the MAC address of the *Access Point* the phone used to register an account, while Device ID is the IMEI/Adsource of the phone used to register an account.

Ethical and privacy considerations: The collected data are already specified in WeChat's privacy policy, which a user consents before using WeChat. Moreover, to protect user privacy, WeChat has anonymized the attributes to the extent such that they are just enough for our analysis. Specifically, an IP address has four segments and each segment is hashed individually. The last four digits (i.e., customer code) of a phone number are removed. All datasets were stored on WeChat's servers and we accessed them through an internship program.



(a) (b)

Figure 1: (a) The number of 24-bit IP prefixes that registered a given number of accounts. (b) The fraction of Sybils among the accounts registered from the 24-bit IP prefixes that registered a given number of accounts.



(a) (b)

Figure 2: (a) The number of phone number prefixes that registered a given number of accounts. (b) The fraction of Sybils among the accounts registered from the phone number prefixes that registered a given number of accounts.

3.2 Synchronization

We find that Sybil accounts exhibit common registration patterns, e.g., they are likely to use the same IPs, phone numbers from the same areas, same devices (identified by their device IDs), and nicknames with the same patterns. We suspect the reason is that an attacker has limited resource (i.e., IPs, phone numbers, and devices) and uses certain scripts to automate registrations of Sybil accounts. With a little abuse of terminology, we call these *synchronized* registration patterns.[2] Next, we describe our measurement results.

IP: WeChat currently only supports registrations from IPv4. An IPv4 address has four segments. We treat the first three segments (i.e., the first 24 bits) as a local network identifier. We acknowledge that, in some scenarios (e.g., CIDR is used), the first three segments may not represent a local network. Nevertheless, we find that using the first three segments to represent an IP is useful for Sybil detection. Therefore, if two IPs have the same 24-bit prefix, then we view them as the same. In total, we have 264,830 24-bit IP prefixes in our dataset. For each 24-bit IP prefix, we group together the benign (or Sybil) accounts that were registered using the IP prefix. Therefore, the size of a group indicates the number of benign (or Sybil) accounts registered from the corresponding 24-bit IP prefix. Figure 1(a) shows the number of IP prefixes that registered a given number of benign (or Sybil) accounts. We observe power-law phenomena for both benign and Sybil registrations, i.e., a majority of IP prefixes registered a small number of accounts, while a small number of IP prefixes registered a large number of accounts. For instance, 34.5% and 15.5% of IP prefixes registered 80% of benign and Sybil accounts, respectively.

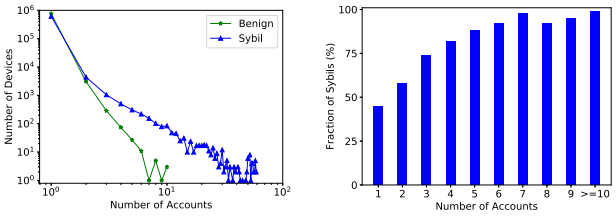
Moreover, Sybil accounts are more likely to be registered from the same IP prefixes. In particular, the IP prefixes that registered a large number of Sybil accounts are much more than those that registered a large number of benign accounts, i.e., the curve corresponding to Sybil is above the curve corresponding to Benign at the tail part in Figure 1(a). Figure 1(b) further shows the fraction of Sybils among the accounts registered from the IP prefixes that registered a given number of accounts. For instance, when the x-axis is 0-1, we find all the IP prefixes that registered 0-500 accounts, and the y-axis is the fraction of Sybils among these accounts. We observe that, when an IP prefix registered a small number of accounts (e.g., 0-500), it is hard to tell whether these accounts are Sybils or not solely based on the fact that they share IP prefix. However, when a large number (e.g., >2,500) of accounts were registered from the same IP prefix, these accounts are more likely to be Sybils.

Phone number: Each phone number can register one account and an attacker could use phone numbers from a certain phone service provider and a certain geographical area to register Sybils. Recall that a phone number except the last four digits indicate a phone service provider and an area code. Therefore, we study the prefix of a phone number with the last four digits removed. Like IP prefix, Figure 2(a) shows the number of phone number prefixes that registered a given number of accounts, while Figure 2(b) shows the fraction of Sybils among the accounts registered from the phone number prefixes that registered a given number of accounts. Like IP prefix, we observe similar synchronization patterns for phone number prefix. In particular, a large number of phone number prefixes registered a small number of accounts, while a small number of phone number prefixes registered a large number of accounts (power-law phenomena); Sybils are likely to be registered in clusters using the same phone number prefixes; and if a phone number prefix registered a large number of accounts (e.g., ≥30), then those accounts are very likely to be Sybils.

Device: Similar to IP prefix and phone number prefix, Figure 3(a) shows the number of devices (identified by IMEI or Adsource) that registered a given number of accounts, while Figure 3(b) shows the fraction of Sybils among the accounts registered from the devices that registered a given number of accounts. We observe similar synchronization patterns: Sybils are likely to be registered from the same devices. An attacker could spoof a device ID, e.g., by modifying the Android application framework. Our results demonstrate that attackers have not performed such device spoofing for the registrations of a large amount of Sybil accounts.

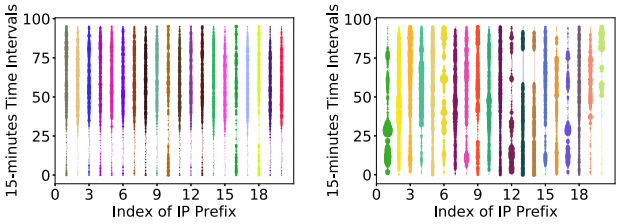
Nickname: Here, we show some qualitative results on nickname pattern analysis. In Section 4.2, we will use natural language processing tools to extract patterns and features from nicknames. Table 2 shows three nickname patterns, an example for each pattern, and the fraction of Sybils among the accounts whose nicknames follow a particular pattern. Note that the nicknames in our dataset are anonymized at the character level by WeChat and we could not

[2]Synchronization typically indicates time synchronization.



(a) (b)

Figure 3: (a) The number of devices that registered a given number of accounts. (b) The fraction of Sybils among the accounts registered from the devices that registered a given number of accounts.



(a) Benign (b) Sybil

Figure 4: The number of accounts registered from an IP prefix in each 15-minutes time interval within a day. Each vertical line represents accounts registered from a certain 24-bit IP prefix, and the size of a dot represents the number of accounts registered in a particular 15-minutes time interval.

Table 2: Nickname pattern.

Pattern	Example	Fraction of Sybils
Chinese+Digits	李四2416	32.2%
Lowercase+Digits	cii2133, vqu7868	94.4%
Digits+Lowercase+Digits	07740922a179	95.0%

understand the semantic patterns in the nicknames. The results in Table 2 were obtained with the help of WeChat security engineers. We observe that some nickname patterns (e.g., Chinese+Digits) are more likely to be used by benign accounts, while some nickname patterns (e.g., Lowercase+Digits, Digits+Lowercase+Digits) are more likely to be used by Sybils. We suspect the reason why Sybils share nickname patterns is that they were registered by scripts, which generate nicknames following certain patterns.

Implications: On one hand, our measurement results indicate that the synchronization patterns are useful to detect Sybils. For instance, we could design a simple method (we will show more details in our experiments in Section 5) to detect Sybil accounts based on the popularity of IP, phone number, or device. Specifically, given all registrations on a single day, we can predict all the accounts registered from an IP prefix (or phone number prefix or device), which registered more than a certain *threshold* of accounts, to be Sybils. For instance, we can predict all the accounts registered from a phone number prefix that registered more than 30 accounts to be Sybils. Such simple detector achieves a precision of almost 100% in our dataset. On the other hand, our results also indicate that synchronization of a single attribute (e.g., IP, phone number, or device) is insufficient to detect Sybils. For instance, the detector based on phone number popularity achieves a recall of only 59%, i.e., 41% of Sybils cannot be detected. Therefore, we will combine 1) synchronization patterns of multiple attributes and 2) anomaly patterns, which we will discuss next.

3.3 Anomaly

Two registrations are synchronized if they share the same IP prefix, phone number prefix, device, or nickname pattern. Our measurement results in the above section demonstrate that Sybils are likely to exhibit synchronization patterns. However, synchronization alone is insufficient to distinguish between benign accounts and Sybils. For instance, suppose two accounts are registered from the same IP prefix that registered a small number of accounts, e.g., 0-500; whether the two accounts are Sybils or not further depends on the risk of the shared IP prefix. In other words, if two accounts are registered from the same IP prefix that is also abnormal, then the two accounts are more likely to be Sybils. Suppose two accounts share a nickname pattern. Table 2 shows that both benign accounts and Sybils share nickname patterns. Therefore, whether the two accounts are Sybils or not further depends on the risk of the shared nickname pattern. Synchronization characterizes whether two accounts share certain registration attributes, while anomaly further characterizes the risk of the shared attributes. Next, we show results on several example anomaly patterns.

Registration time: Figure 4 shows the distribution of accounts registered from the same IP prefix with respect to time. Each graph shows results for 20 example IP prefixes. We divide the 24 hours in a day into 96 15-minutes intervals. Each vertical line in a graph shows the accounts registered from a certain IP prefix in these 96 time intervals, where the size of a dot is proportional to the number of accounts registered in the corresponding time interval. We observe that when a large amount of benign accounts were registered from the same IP prefix, they were evenly registered in the daytime and a small number of them were registered during midnight. However, when many Sybils were registered from the same IP prefix, they were registered in clusters and during late night. Note that we normalized the times with respect to their time zones determined by their IPs.

Geolocation inconsistency: An IP address can be mapped to a geolocation, and a phone number can also be mapped to a geolocation using its area code. We find that 65% of Sybils have different IP-based location and phone-number-based location at the time of registration. A possible reason is that attackers use remote compromised machines or cloud servers and phone numbers obtained from local areas to register Sybils. Another possible reason is that attackers buy phone numbers from other areas [36] and use local machines to register Sybils. Moreover, a user can specify its location (e.g., country) as a part of its account profile when registering an

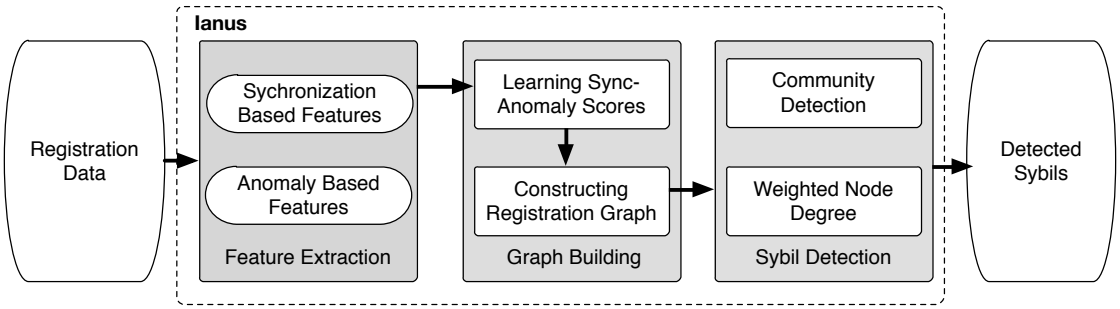


Figure 5: Overview of Ianus

account. We find that 96% of Sybils specified countries that are inconsistent with the IP-based ones. A possible reason is that these Sybils aim to target benign users from particular locations. We note that such geolocation inconsistency was also observed in dating fraud [15].

Rare and outdated WeChat and OS versions: We observe that accounts registered from rare and outdated WeChat and OS versions are more likely to be Sybils. We say a WeChat or OS version is rare if the number of accounts registered from it is small (i.e., less than 3%). For instance, a certain Android version only registered 2K accounts in our dataset and 96.5% of them are Sybils. Likewise, 99% of accounts registered from iOS 8 (an outdated OS version) are Sybils. Possible reasons include that attackers use old devices with outdated WeChat and OS versions to register Sybils and that attackers automatically register Sybils using scripts and have not updated their WeChat and OS versions in the scripts.

4 DESIGN OF IANUS

4.1 Overview

Ianus aims to detect Sybils using their registration data. Ianus consists of three key components, i.e., *feature extraction*, *graph building*, and *Sybil detection*. Figure 5 shows the three components. In feature extraction, we extract features for a *pair* of registrations. Inspired by our measurements on synchronization and anomaly patterns in the previous section, we extract *synchronization based features* and *anomaly based features*. Synchronization based features characterize whether a pair of registrations have the same registration attributes (e.g., IP prefix, phone number prefix, device ID), while anomaly based features further characterize whether these attributes are abnormal.

The graph building component aims to construct a weighted graph to integrate the heterogeneous synchronization and anomaly patterns. We call the graph *registration graph*. A node in the graph is a registration. We aim to build a registration graph such that Sybils are densely connected with each other by edges with large weights, while benign accounts are sparsely connected with each other and Sybils. To build such a graph, we first learn a *sync-anomaly score* for each pair of registrations using a logistic regression classifier. We use the sync-anomaly score to quantify the synchronized and anomaly patterns between two registrations. A higher sync-anomaly score means that the two registrations are more likely to be Sybils. Then, we create an edge between two registrations if their sync-anomaly score is large enough when building the registration graph.

The Sybil detection component aims to detect Sybils via analyzing the structure of the registration graph. For instance, community detection is a natural choice to detect Sybils since Sybils are densely connected in our registration graph but benign accounts are not. Moreover, we propose a simple weighted node degree based method to detect Sybils in the registration graph, which achieves similar accuracies with the community detection method but is much faster. In our registration graph, a node that has a higher weighted degree is more likely to be Sybil. Therefore, we predict a node to be Sybil if its weighted degree is larger than a threshold, which is learnt using machine learning techniques.

4.2 Extracting Features

We define features for a pair of registrations. Specifically, our measurement results in Section 3 indicate that Sybils demonstrate both synchronized and abnormal registration patterns. Therefore, we extract two categories of features, i.e., *synchronization based features* and *anomaly based features*. Synchronization based features characterize what attributes (e.g., IP, phone number, device) are shared by the two registrations, while anomaly based features further characterize whether the shared attributes or the two registrations are abnormal. All of our features are binary.

4.2.1 Synchronization Based Features. Table 3 shows our synchronization based features. We add a prefix "S-" to each feature to indicate that they are synchronization based features. Most of these features are self-explained. For instance, the feature S-IP24 is 1 if and only if the two registrations use IPs with the same 24-bit prefix. S-PN is 1 if and only if the two registrations use the same phone number prefix, where the last four digits of a phone number are removed. Recall that, in China, a phone number prefix without the last four digits indicates the phone service provider and location where the number is obtained from. Therefore, using the same phone number prefix could indicate that the two accounts were registered in clusters by an attacker. The nickname based features (S-NP1 and S-NP2) are the most challenging features to compute because users can specify their nicknames arbitrarily. We leverage natural language processing techniques to extract patterns from nicknames and compute these features. The feature S-IP32 was also used by EvilCohort [34]. S-IP24, S-NP1, and S-NP2 were used by

Table 3: Synchronization based features. All features are binary. If two registrations have the same value for the corresponding attribute, the corresponding feature is 1, otherwise the feature is 0.

Feature	Description (Both registrations...)
S-IP24	use the same 24-bit IP prefix
S-IP32	use the same IP address
S-PN	use the same phone number prefix
S-OS	use the same OS version
S-WeChat	use the same WeChat version
S-MAC	use the same WiFi MAC address
S-Device	use the same device
S-NP1	have the same syntactic nickname pattern
S-NP2	have the same semantic nickname pattern

Thomas et al. [38]. Next, we discuss details on extracting nickname based features.

Nickname based features: We extract a *syntactic pattern* and a *semantic pattern* from a nickname. The features S-NP1 and S-NP2 are based on the syntactic and semantic patterns, respectively. We define a syntactic pattern as a string of characters from the vocabulary $V = \{C, L, U, D, \cdots\}$, where C represents Chinese characters, L and U represent lowercase and uppercase English letters, respectively, D represents digits, while each punctuation and special character (e.g., ;, ., +) is still a character in the vocabulary. For instance, the syntactic pattern of the nickname 李雷abAB12++ is *CCLLUUDD++*. The feature S-NP1 is 1 for a pair of registrations if their nicknames have the same syntactic pattern.

The syntactic pattern considers the structure of a nickname but ignores its semantics. For example, a random Chinese string and a Chinese name could have the same syntactic pattern but different semantics. Two accounts are more synchronized if their nicknames are both random Chinese strings, as random Chinese strings may be automatically generated by attackers' scripts. Therefore, we further extract semantic patterns from nicknames. Specifically, we define several semantic patterns including *Chinese phrase*, *random Chinese string*, *English phrase*, *Chinese pinyin*, and *random English string*. We extract the semantic patterns Chinese phrase and random Chinese string for nicknames that use only Chinese characters, and we extract the semantic patterns English phrase, Chinese pinyin, and random English string for nicknames that use only English letters. Chinese pinyin is a way to represent Chinese characters using English letters.

To extract these semantic patterns, we collected a large Chinese corpus consisting of hundreds of thousands of articles from WeChat subscription and a large English corpus including hundreds of thousands of English words and millions of Chinese pinyin. Then, we trained n-gram models separately for Chinese sentences, English sentences, and Chinese pinyin using the tools called srilm [32] and Jieba (a tool to segment Chinese sentences) [1]. Finally, we use the n-gram models to extract semantic patterns for nicknames. Specifically, if a nickname is Chinese string that has a large probability and a small perplexity under the n-gram model (we use the thresholds 10^{-15} and 10,000 for the probability and perplexity, respectively), then we say the nickname is a Chinese phrase, otherwise it is a random Chinese string. Likewise, an English nickname is English phrase or Chinese pinyin if it has a large probability and a small perplexity under the corresponding n-gram models, otherwise it is a random English string. The feature S-NP2 for a pair of registrations is 1 if their nicknames have the same semantic pattern.

Table 4: Anomaly based features. These features are also binary. A feature is 1 if the corresponding attribute of both registrations is abnormal.

Feature	Description (Both registrations...)
A-Location	have different user-specified and IP-based countries
A-OS	use rare or old OS versions
A-WeChat	use rare or old WeChat versions
A-Time	were registered at late night, i.e., 2am–5am
A-NP	have the same nickname pattern that is abnormal

4.2.2 Anomaly Based Features. We also extract anomaly based features for a pair of registrations. These anomaly based features characterize whether a pair of registrations both have abnormal attributes. Intuitively, anomaly can be used to characterize the attributes of each individual registration. Therefore, anomaly based features can be extracted for each individual registration, and the features for a pair of registrations can be concatenated as features for them. More formally, we could extract anomaly based features F_A for registration A. Then, for a pair of registrations A and B, we can concatenate their features (F_A, F_B) as the unified features for them. However, such concatenated feature is significantly influenced by the ordering of the two registrations (i.e., (F_A, F_B) vs. (F_B, F_A)), and there is no canonical ordering for a pair of registrations. As a result, it is much harder to learn the sync-anomaly score for a pair of registrations when building the registration graph. Therefore, we jointly consider both registrations when extracting anomaly based features, which does not depend on the ordering of the two registrations.

Table 4 shows our anomaly based binary features. We add a prefix "A-" to them to indicate that they are anomaly based features. Next, we describe details of these features.

Geolocation inconsistency (A-Location): In WeChat, a user can arbitrarily specify its location, e.g., country, in its profile. This location information will be displayed to the user's friends. Moreover, when a user A sends a friend request to another user B, user B can view some basic information (including location) in user A's profile. Therefore, some Sybils specify their locations as a particular location with a goal to target users in that location, no matter where the Sybils were really registered from. Note that we can use IP address to determine the location where an account was registered from. An account has geolocation inconsistency if its user-specified location and IP-based location are different. Since our anomaly based features characterize the abnormal patterns shared by a pair of registrations, we define a binary feature A-Location for a pair of registrations, which is 1 if both registrations have geolocation

inconsistency. We consider location at the level of country in our experiments.

Rare and outdated OS and WeChat versions (A-OS and A-WeChat): We design two features for OS and WeChat versions, respectively. We compute the fraction of registrations that use a particular OS version. If a small fraction of registrations use an OS version, we say the OS version is rare. In our experiments, we say an OS version is rare if less than 5% of registrations use it. Moreover, we manually label some OS versions as outdated, according to domain knowledge. For instance, we label the iOS versions lower than version 8 as outdated. We view a registration as abnormal if it uses a rare or outdated OS version, and the binary feature A-OS for a pair of registrations is 1 if both registrations use rare or outdated OS versions. Similarly, we define a binary feature A-WeChat for a pair of registrations, which is 1 if both registrations use rare or outdated WeChat versions.

Registration time (A-Time): Our measurement results in Figure 4 show that Sybil accounts were registered even at late night, while benign accounts are mainly registered at daytime. Therefore, we design a binary feature A-Time, which is 1 if and only if both accounts were registered at late night, i.e., between 2am and 5am in our work.

Nickname pattern (A-NP): As we discussed in Section 4.2.1, we extract syntactic and semantic nickname patterns using natural language processing techniques. Example semantic nickname patterns include Chinese phrase, random Chinese string, English phrase, Chinese pinyin, and random English string. Intuitively, random Chinese strings (e.g., 鲍技坦痹) and random English strings (e.g., *nzadnhen*) are rarely used by normal users since they are not meaningful. Therefore, we treat these as abnormal nickname patterns. If two registrations share a nickname pattern and the nickname pattern is abnormal, then their binary feature A-NP is 1.

4.3 Building a Registration Graph

We aim to construct a weighted graph to represent the relationships between accounts, where a node is an account, Sybils are more likely to be connected with each other by edges with large weights, and benign accounts are more likely to be sparsely connected. We call this graph *registration graph* since it is built using registration information. We first leverage machine learning techniques to learn a score for each pair of registrations using their feature vector. A larger score means that the pair of registrations share more (abnormal) attributes and are more likely to be Sybils. We call the score *sync-anomaly score*. Then, we construct a registration graph based on the sync-anomaly scores.

4.3.1 Learning Sync-anomaly Score. We derive a sync-anomaly score for a pair of registrations to quantify their synchronization and anomaly patterns. Our goal is to compute sync-anomaly scores such that a pair of registrations have a large sync-anomaly score if they both are Sybils but a small sync-anomaly score if they both are benign. A key challenge to compute such scores is that different synchronization based features and anomaly based features could have different influences on the scores.

To address the challenge, we leverage *supervised machine learning* to automatically learn the weights of different features using historical data. Specifically, we construct a training dataset using historical data, which includes 1) a set of registrations and 2) their labels (i.e., benign or Sybil). Given the training dataset, we compute the features for each pair of registrations. Moreover, we derive a label for each pair of registrations, where the label Positive means that both registrations are Sybils and the label Negative means that both registrations are benign. Given the features and labels for each pair of registrations in the training dataset, we learn a logistic regression classifier and use its output as a sync-anomaly score.

Assigning labels for pairs of registrations: Given a training dataset, we first compute the feature vector for each pair of registrations. Recall that all our features are binary. The labels of a pair of registrations can be (0, 0), (0, 1), (1, 0), or (1, 1), where 0 and 1 represent benign and Sybil, respectively. A naive method to derive a label (Positive or Negative) for a pair of registrations is simply to use the labels of the two individual registrations. Specifically, a pair of registrations has a Positive label if both of them are Sybils and a Negative label if both of them are benign. However, such naive method could assign inconsistent labels to the same feature vector. Specifically, two pairs of registrations could have the same feature vector but are assigned different labels, e.g., due to randomness in features and inaccurate labels for individual registrations. Such inconsistent labels make it much harder (if possible) to learn the sync-anomaly scores using supervised machine learning.

Therefore, we assign a label to a binary feature vector via considering all pairs of registrations in the training dataset. For instance, a feature vector has a Positive label if a majority of pairs of registrations that have this feature vector are Sybils. However, such method faces another challenge, i.e., some feature vectors only appear in a very small number of pairs of registrations. To address the challenge, we consider features individually in a feature vector. Specifically, suppose we have a feature vector f_a. We denote by $S(f_a)$ and $T(f_a)$ the set of pairs of Sybils and the set of all pairs of registrations whose feature vector is f_a. We call $S(f_a)$ and $T(f_a)$ *Sybil support* and *support* of the feature vector f_a, respectively. For any other feature vector f_b, if the features that are 1 in F_a are also 1 in f_b (i.e., feature vector f_b includes f_a), then we expand the Sybil support and support of feature vector f_a to include those of f_b, i.e., $S(f_a) = S(f_a) \cup S(f_b)$ and $T(f_a) = T(f_a) \cup T(f_b)$.

Finally, we compute a *Sybil support ratio* of feature vector f_a as $|S(f_a)|/|T(f_a)|$. If the Sybil support ratio of a feature vector is larger than a *threshold*, then we assign a Positive label to the feature vector, otherwise the feature vector has a Negative label. We choose a large threshold (i.e., 0.98) to enforce a high standard of assigning a Positive label for a pair of registrations. Recall that the labels for individual registrations could be incorrect. Therefore, we select a threshold of 0.98 instead of 1.00 to consider such label mistakes. Our goal is to construct a graph in which Sybils are more likely to be densely connected. This higher standard of assigning a Positive label makes it harder for two accounts that are not both Sybils to be connected in our graph. We study the impact of different thresholds on Ianus in our experiments.

Learning sync-anomaly scores: After assigning labels to feature vectors, we have a set of pairs (feature vector, label), which we use to learn a sync-anomaly score for a pair of registrations. Specifically, given the pairs (feature vector, label), we learn a binary logistic

regression classifier. The classifier takes a feature vector as an input and outputs the probability that the feature vector has a Positive label. After learning the logistic regression classifier using a historical training dataset, we can apply the classifier to pairs of registrations in the future. Specifically, for a pair of registrations, we construct its feature vector and use the classifier to compute the probability that the feature vector has a Positive label (i.e., both registrations are Sybils). We treat such probability as the sync-anomaly score for the pair of registrations. Our sync-anomaly score ranges from 0 to 1.

Note that the learnt logistic regression classifier can also be used to predict whether a pair of registrations are both Sybils or not. Specifically, in a binary logistic regression classifier, an instance is often predicted to be Positive if its probability of being Positive is larger than 0.5. Therefore, if a pair of registrations have a sync-anomaly score larger than 0.5, the logistic regression classifier predicts that both of them are Sybils. We will leverage such classifier to prune edges when constructing the graph.

4.3.2 Constructing a Registration Graph. Our goal is to construct a registration graph in which an account is a node, Sybils are more likely to be densely connected with each other by edges with large weights, and benign accounts are more likely to be isolated. Towards this goal, we create an edge between two accounts only if both of them are predicted to be Sybils. Recall that a sync-anomaly score that is larger than 0.5 means that the corresponding two accounts are both predicted to be Sybils. Therefore, we create an edge between two accounts only if their sync-anomaly score is larger than 0.5, and we use the sync-anomaly score as the edge weight.

A naive method to construct the registration graph is to compute the sync-anomaly score for each pair of registrations/accounts and create edges based on the sync-anomaly scores. However, such pairwise comparison is not scalable to a large number of registrations. For instance, WeChat has millions of registrations in a week. If we plan to detect Sybils on a weekly basis, we need to compute over 10^{14} sync-anomaly scores to construct the registration graph, which is very challenging if possible.

We propose techniques to address this computational challenge. Our key intuition is that if a pair of registrations are both Sybils, then they are very likely to have synchronized attributes, e.g., they have the same IP prefix, phone number prefix, or device ID. Therefore, for each registration attribute including the 24-bit IP prefix, phone number prefix, and device ID, we divide registrations into groups, where each group has the same value for the considered registration attribute. Then, we only compute sync-anomaly scores for pairs of registrations within the same group and create edges if the sync-anomaly scores are larger than 0.5. Note that no matter how many groups a pair of registrations appear in, we only need to compute their sync-anomaly score once as the score does not depend on how many groups the pair of registrations appear in.

4.4 Detecting Sybils

Community detection: In our constructed registration graph, Sybils are densely connected with each other, while benign accounts are sparsely connected with each other and Sybils. Therefore, community detection is a natural choice to detect Sybils in the registration graph. For instance, we can use the popular Louvain method [5] to detect communities in the registration graph and predict accounts in the communities whose sizes are larger than a threshold to be Sybils. Note that the Louvain method was also used by EvilCohort [34] to detect Sybils based on user logins.

Table 5: Dataset statistics.

	#Sybils	#Benign
Dataset I	779k	681k
Dataset II	647k	770k

Weighted node degree: We also propose a simple weighted node degree based method to detect Sybils in our registration graph. The weighted degree of a node is the sum of the weights of the node's edges. Intuitively, if a node is connected with more neighbors in the registration graph, then the node is more likely to be Sybil. This is because, for each neighbor of the node, the neighbor and the node are predicted to be both Sybils using their sync-anomaly score (i.e., edge weight), according to how we build the registration graph. Therefore, a node with a larger weighted node degree could be more likely to be Sybil.

We learn a binary classifier to detect Sybils based on the weighted node degree. The classifier takes a node's weighted degree as a feature input and predicts whether it is a Sybil or not. We find that weighted node degrees span a very wide range of values, making detection inaccurate. Therefore, we first normalize them using the *tanh* function (output is in the interval (-1, 1)), which acts as feature normalization. Specifically, for a node, we put the node's weighted degree into the *tanh* function, and we treat the function output as the node's normalized weighted degree. Given a training dataset consisting of both labeled benign nodes and labeled Sybil nodes (e.g., the training dataset we use to learn the sync-anomaly scores in Section 4.3.1), we learn a binary classifier to detect Sybils. While any binary classifier is applicable, we leverage an ensemble classifier called *EasyEnsemble* [53] in our experiments to cope with the imbalanced training dataset. Although our dataset is not highly imbalanced (see Table 5), we found that EasyEnsemble still improves detection accuracy upon simple classifiers like logistic regression.

We find that our weighted node degree based method and the Louvain community detection method achieve close accuracies (see our experimental results in Section 5), but our weighted node degree based method is much faster than the Louvain method.

5 EVALUATION

We evaluate Ianus using datasets from WeChat. Moreover, WeChat has deployed Ianus to detect Sybils in the wild. We first describe our experimental setup. Then, we show detection results of Ianus and its different variants using labeled datasets. Finally, we discuss results of real-world deployment of Ianus at WeChat.

5.1 Experimental Setup

Datasets: We obtained two registration datasets collected in October and November, 2017 from WeChat. *Dataset I* includes registrations in October, while *Dataset II* includes registrations in November that is around one week *after* Dataset I. Our measurement results

in Section 3 were obtained using Dataset II. Table 5 shows the number of Sybils and benign accounts in each dataset. The labels were provided by the security team of WeChat, who verified that the labels have an accuracy larger than 95%. We will use these labels as ground truth for evaluation. In particular, the labels were obtained in the same way as we described in Section 3.1.

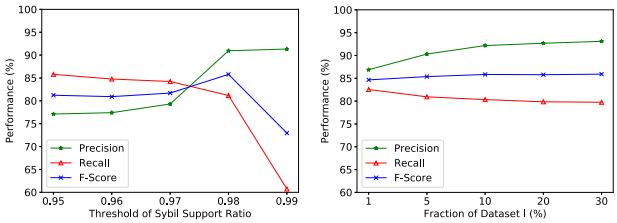
Training and testing: Our Ianus requires a training dataset to learn the sync-anomaly scores in the graph building component and learn the node degree based classifiers in the Sybil detection component. To simulate real-world scenarios, we construct a training dataset from *historical registrations* and detect Sybils in future registrations. In particular, we sample some registrations in Dataset I and treat them as a training dataset, while testing Ianus on Dataset II. We do not use all registrations in Dataset I as a training dataset because it may be challenging to label all registrations from one day in practice. Moreover, we explored the performance of Ianus when different fractions of Dataset I are used as the training dataset (results are shown in Figure 6(b)). We found that the performance stabilizes after 10% of Dataset I are used as the training dataset. Therefore, without otherwise mentioned, we sample 10% of registrations in Dataset I as a training dataset.

Compared methods: We will evaluate different variants of Ianus and compare Ianus with popularity-based methods.

Variants of Ianus. We consider the following variants:

- **Ianus.** Ianus uses both synchronization and anomaly based features; Ianus uses logistic regression to learn sync-anomaly scores; and Ianus uses the weighted node degree based method to detect Sybils. Specifically, we used the logistic regression implemented in Spark with the default parameter setting.
- **Ianus-Sync and Ianus-Anomaly.** Ianus-Sync and Ianus-Anomaly use synchronization and anomaly based features, respectively.
- **Ianus-FS.** Ianus-FS sums the binary features as the sync-anomaly score for a pair of registrations and creates an edge between two registrations if the sync-anomaly score is larger than a threshold. We will study the impact of different thresholds on Ianus-FS.
- **Ianus-CD.** Ianus-CD uses the Louvain method to detect communities in the registration graph and treats the communities whose sizes are larger than a threshold as Sybils. We will study different thresholds.
- **Ianus-FS-CD.** This variant combines Ianus-FS and Ianus-CD. Specifically, Ianus-FS-CD sums the binary features as the sync-anomaly scores in the graph building component and uses community detection in the Sybil detection component. Note that Ianus-FS-CD is an *unsupervised* method, as it does not require a historical training dataset to learn the sync-anomaly scores and the node degree based classifiers.

Popularity-based methods. Our measurement results in Section 3.2 inspire us to design simple popularity based methods to detect Sybils. Specifically, for an attribute value (e.g., phone number prefix, device ID), we compute its popularity as the number of registrations in the testing dataset that use the attribute value. If an attribute value has a popularity larger than a threshold, we predict all registrations that have the attribute value to be Sybils.



(a) (b)

Figure 6: (a) Impact of the threshold of Sybil support ratio on Ianus. (b) Impact of the training dataset size on Ianus.

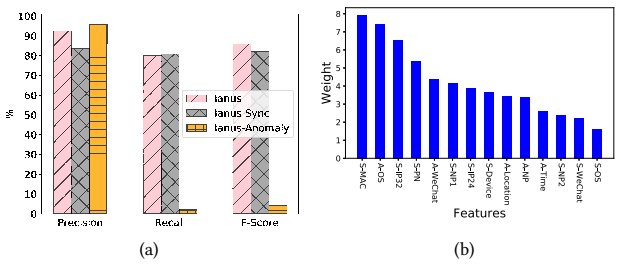
Evaluation metrics: We use standard metrics in information retrieval, i.e., *Precision*, *Recall*, and *F-Score*, to evaluate performance. Specifically, for a method, Precision is the fraction of its predicted Sybils that are true Sybils in the testing dataset, Recall is the fraction of true Sybils in the testing dataset that are predicted as Sybils by the method, and F-Score is harmonic mean of Precision and Recall.

5.2 Results

Ianus is effective: On Dataset II, Ianus achieves Precision 92.4%, Recall 80.2%, and F-Score 85.9%, respectively. A key reason why Ianus is effective is that Sybils are densely connected in our constructed registration graph but benign accounts are not. Specifically, in our registration graph, on average, a Sybil is connected with 280 other Sybils, a Sybil is connected with only 1.3 benign nodes, and a benign node is connected with only 4.5 other benign nodes.

In our registration graph, we find that about 69% of false positives are in communities (detected by the Louvain's method) with sizes less than 10 and 7% of false positives are in communities with sizes between 10 and 100. The main reason of these false positives is that their features such as phone number based and IP based ones are similar to those of the Sybils. Moreover, we observed that 3% of false positives are in communities with sizes between 100 and 1,000 and 21% of false positives are in communities with sizes between 1,000 and 10,000. The main reason of these false positives is that they were registered from the same organizations and share similar patterns.

Impact of the threshold of Sybil support ratio: Recall that Ianus leverages a threshold of Sybil support ratio to assign labels to feature vectors when learning the sync-anomaly scores. A natural question is how this threshold impacts the accuracy of Ianus. Figure 6(a) shows the detection results of Ianus as a function of the threshold. We observe that Precision increases and Recall decreases when the threshold increases from 0.95 to 0.98. The reason is that a larger threshold makes it harder for two registrations to be connected in our registration graph. On one hand, the connected nodes in the registration graph are more likely to be Sybils when a higher threshold is used, and thus Ianus can detect Sybils with a higher Precision. On the other hand, more Sybils are disconnected or sparsely connected in the registration graph, resulting in a lower Recall. However, when the threshold increases from 0.98 to 0.99,



(a) (b)

Figure 7: a Comparing Ianus, Ianus-Sync, and Ianus-Anomaly. (b) Weights of features in the logistic regression classifier that learns the sync-anomaly scores.

Precision just increases slightly but Recall drops significantly, resulting in a significant drop in F-Score. Therefore, we choose 0.98 as a threshold.

Impact of the training dataset size: Figure 6(b) shows the detection results of Ianus when different fractions of Dataset I are used as the training dataset. When increasing the training dataset size from 1% to 10%, the Precision increases and the Recall decreases. This indicates that, with more training data, connected nodes in the registration graph are more likely to be Sybils, but less nodes are densely connected. However, the Precision and Recall start to stabilize after 10% of training dataset size. This implies that Ianus does not need a very large training dataset, and a relatively small training dataset suffices. Ianus uses 10% training dataset size to achieve a high Precision. WeChat may suspend the detected Sybils. A higher Precision means that less benign users would need to reactivate their suspended accounts via a particular procedure (e.g., calling the service center of WeChat).

Ianus vs. Ianus-Sync and Ianus-Anomaly: Figure 7(a) shows the detection results of Ianus, Ianus-Sync, and Ianus-Anomaly. Ianus outperforms Ianus-Sync and Ianus-Anomaly. Our results indicate that combining synchronization based features and anomaly based features does improve detection results. This is because synchronization based features and anomaly based features are complementary to each other. Specifically, synchronization based features characterize whether two accounts share common registration attributes, while anomaly based features further characterize whether two accounts share abnormal registration attributes. We note that Ianus-Anomaly achieves a high Precision but a very low Recall. This is because a small number of Sybils have significant abnormal registration patterns, which distinguish them with other Sybils and benign accounts; and Ianus-Anomaly can only detect these Sybils.

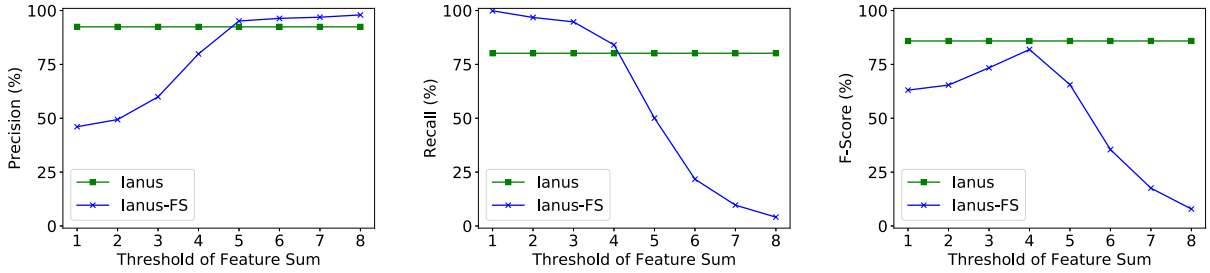
Feature weights: Figure 7(b) shows the weights of features in the logistic regression classifier that we use to learn the sync-anomaly scores. All features have positive weights, which means that all features have positive impact on the sync-anomaly scores and subsequently on the overall performance of Ianus. When Ianus is deployed and known by attackers, they may adjust their strategies to evade Ianus, e.g., by adjusting registration features or providing fake ones. Indeed, some of our features can be spoofed by attackers with low costs. For instance, an attacker could modify the OS version based features (S-OS, A-OS), the WeChat version based features (S-WeChat, A-WeChat), the device based feature (S-Device), the WiFi MAC based feature (S-MAC), the registration time based feature (A-Time), and the location based feature (A-Location) with a low cost, because the attacker could use appropriate OS and WeChat versions, spoof device ID and WiFi MAC, register during daytime, as well as specify a location that is consistent with the IP-based location (the attacker can still modify the specified location at any time to target benign users in a particular area).

However, some features incur larger costs for attackers to modify. For instance, to modify the IP based features (S-IP32, S-IP24) and the phone number based feature (S-PN), an attacker needs more diverse real IPs and phone numbers, which incurs larger economic costs. Moreover, to spoof the nickname based features (S-NP1, S-NP2, A-NP), an attacker needs sophisticated natural language processing techniques. Some of these harder-to-evade features have large weights, which shows the potential robustness of Ianus against evasion. Indeed, if we only use the harder-to-evade features (i.e., S-IP32, S-IP24, S-PN, S-NP1, S-NP2, A-NP), Ianus achieves a Precision of 93.6% and a Recall of 45.1%, i.e., Ianus can still accurately detect Sybils, though it can detect less Sybils.

Ianus vs. Ianus-FS: Ianus-FS uses feature sum to compute sync-anomaly scores for pairs of registrations and creates an edge between two registrations if their sync-anomaly score is larger than a threshold. We study different thresholds. Figure 8 shows the results of Ianus-FS with different thresholds. We observe that, as the threshold increases, the Precision increases and Recall decreases. The reason is that a higher threshold sets a higher standard for creating an edge between two registrations, which makes less nodes connected (lower Recall) but the connected nodes more likely to be Sybils (higher Precision) in the registration graph. Ianus-FS achieves the largest F-Score when the threshold is 4, which is around 4% lower than that of Ianus.

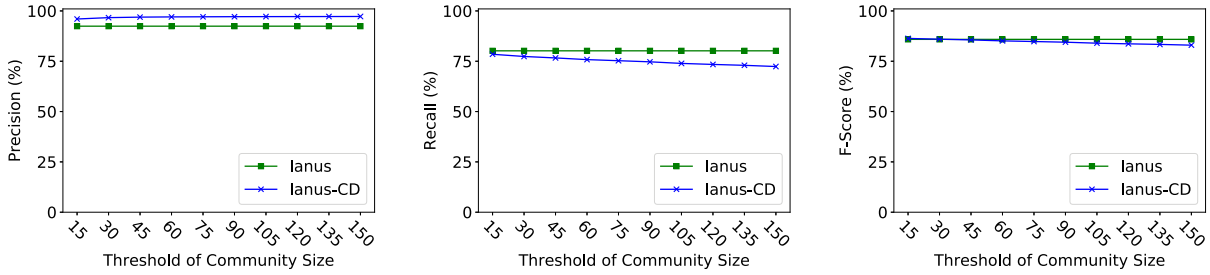
Ianus vs. Ianus-CD: Ianus-CD uses the Louvain method to detect communities and predicts nodes in the communities whose sizes are larger than a threshold as Sybils. Figure 9 shows the results of Ianus-CD for different thresholds. Compared to Ianus, Ianus-CD has a higher Precision, a lower Recall, and no larger F-Score. Ianus-CD has the same F-Score with Ianus when the threshold is 15 or 30. However, Ianus is much faster than Ianus-CD. Specifically, given the registration graph, the Louvain method takes 40 minutes, while our node degree based method takes only 10 minutes on the same platform.

Ianus vs. Ianus-FS-CD: Ianus-FS-CD has two key thresholds, i.e., the threshold of feature sum to determine whether creating an edge between two accounts and the threshold of community size to determine which communities are Sybils. Figure 10 shows the results of Ianus-FS-CD vs. the threshold of community size, where the threshold of feature sum is set to be 4 according to the results in Figure 8. Overall, Ianus-FS-CD achieves a lower F-Score than Ianus. Specifically, the F-Score of Ianus-FS-CD is around 3.5% lower than that of Ianus. This is because the logistic regression based sync-anomaly scores outperform the feature sum based ones. We also find that, when fixing the threshold of community size (e.g., 30), the performance of Ianus-FS-CD vs. the threshold of feature sum has similar patterns with those of Ianus-FS in Figure 8. For



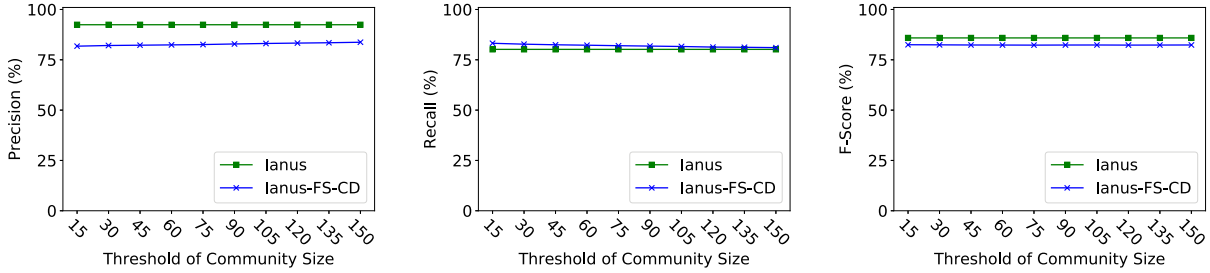
(a) Precision (b) Recall (c) F-Score

Figure 8: Comparing Ianus with Ianus-FS.



(a) Precision (b) Recall (c) F-Score

Figure 9: Comparing Ianus with Ianus-CD.

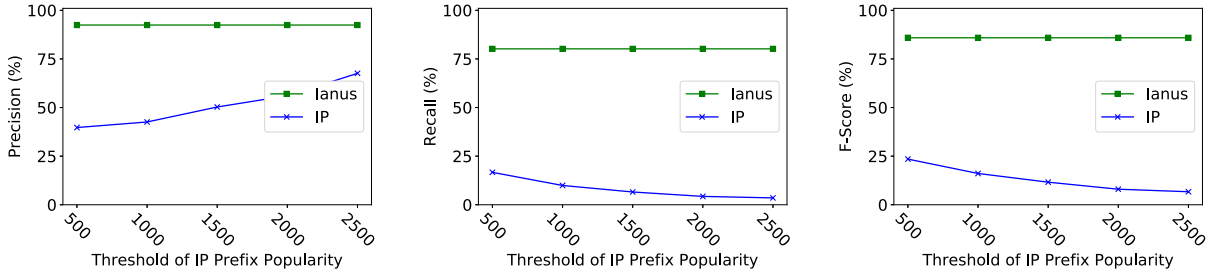


(a) Precision (b) Recall (c) F-Score

Figure 10: Comparing Ianus with Ianus-FS-CD, where the threshold of feature sum is 4.

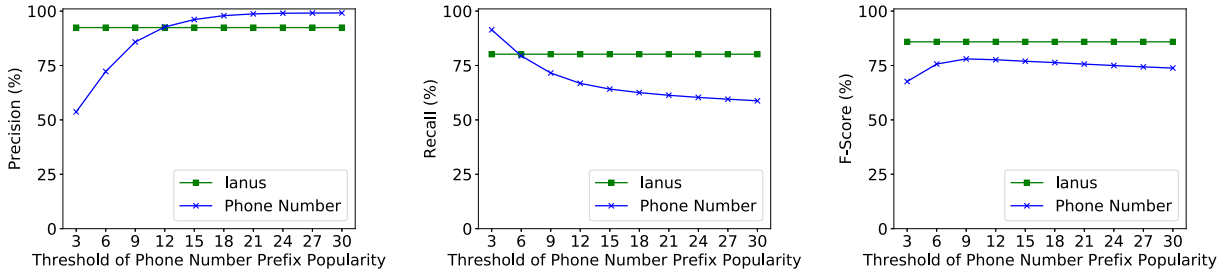
instance, Ianus-FS-CD also achieves the highest F-Score when the threshold of feature sum is 4, which is still around 4% lower than that of Ianus. For simplicity, we omit the results of Ianus-FS-CD with respect to the threshold of feature sum.

Ianus vs. popularity-based methods: We compare three popularity-based methods using IP prefix, phone number prefix, and device ID. Such popularity-based methods are less effective for other registration attributes such as nickname patterns, and thus we omit their results. If the number of registrations from a 24-bit IP prefix (or phone number prefix or device ID) in the testing dataset is more than a *threshold*, the popularity-based method predicts all these registrations to be Sybils. Figure 11, 12, and 13 show the results of popularity-based methods that use IP prefix, phone number prefix, and device ID, respectively. We find that the IP popularity based method is much less effective than the popularity-based methods that use phone number and device ID with respect to Precision. The reason is that a large amount of benign accounts are also registered from IPs with the same prefixes. As the popularity threshold increases, the popularity-based methods that use phone number prefix and device ID can have larger Precision than Ianus (Precision can even reach 100%), but the Recall is much lower than Ianus with such thresholds.



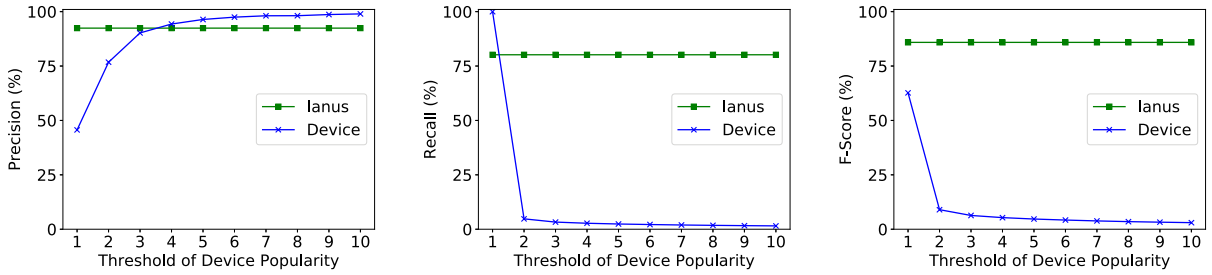
(a) Precision (b) Recall (c) F-Score

Figure 11: Comparing Ianus with a popularity-based method that uses 24-bit IP prefix.



(a) Precision (b) Recall (c) F-Score

Figure 12: Comparing Ianus with a popularity-based method that uses phone number prefix.



(a) Precision (b) Recall (c) F-Score

Figure 13: Comparing Ianus with a popularity-based method that uses device ID.

Table 6: Combining Ianus and popularity-based methods.

	Precision	Recall	F-Score
Phone-Device	98.0%	68.8%	80.8%
Ianus	92.4%	80.2%	85.9%
Ianus + Phone-Device	92.2%	82.6%	87.1%

We can also combine phone number prefix and device ID as a unified popularity-based method. Specifically, we take the union of the predicted Sybils of phone number and device based methods as the final predicted Sybils of the unified popularity-based method. We denote the unified method as *Phone-Device* based method. Moreover, we can further combine Ianus with the Phone-Device based method, i.e., we take the union of their predicted Sybils as the predicted Sybils. Table 6 shows the detection results of Phone-Device, Ianus, and Ianus + Phone-Device, where the popularity thresholds of phone number prefix and device are chosen as 21 and 4, respectively. We choose these thresholds such that individual phone number and device based method achieves a very high Precision and a relatively large Recall.

Table 7: Number of accounts that Ianus predicts to be Sybils per million new registered accounts every day in a certain week after being deployed at WeChat. The Precision is 96% on average.

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
#Detected Sybils	434K	477K	454K	372K	377K	327K	295K

We find that combining Ianus with the popularity-based methods can detect more Sybils (i.e., higher Recall) without sacrificing the Precision of Ianus. On one hand, our results show that some Sybils can be detected by both Ianus and the popularity-based methods. On the other hand, Ianus and the popularity-based methods are complementary to each other, i.e., some Sybils detected by Ianus are not detected by the popularity-based methods and vice versa.

5.3 Real-world Deployment at WeChat

Implementation of Ianus: We implemented Ianus on Spark with scala and Python. We use the Jieba and n-gram training tool srilm [32] to analyze nickname patterns. Moreover, we leverage GraphX and MLlib on Spark to process graphs and implement machine learning classifiers.

Detection results: WeChat deploys Ianus to detect Sybils on a daily basis. Specifically, on each day, WeChat uses Ianus to analyze the accounts registered in the previous day and detect Sybils. Ianus detects around 400K Sybils per million new registered accounts every day in the wild. Table 7 shows the number of accounts that Ianus predicts to be Sybils every day in a certain week in three months after we trained Ianus. The WeChat security team manually inspected 40K accounts sampled uniformly at random from the accounts registered during this period. Specifically, they inspected an account's public information such as profile picture, moments/posts, friend invitations, etc.. The WeChat security team adopted a conservative criteria when labeling an account as Sybil. In particular, an account is labeled as Sybil if the account posts a large number of ads, sends friend requests frequently, posts pornography content, etc. 51.1% of the 40K accounts were labeled as Sybils and the remaining 48.9% were labeled as benign. Ianus predicted 39.9% of the 40K accounts to be Sybils. Moreover, Ianus achieves Precision 96% and Recall 75%. WeChat runs Ianus alongside its other existing detection systems and suspends the accounts that are predicted as Sybils by multiple systems. Some benign accounts may also be incorrectly suspended. However, users can apply to reactivate their suspended accounts via a manual verification process. Overall, around 6% of suspended accounts applied to unlock their accounts.

6 DISCUSSION AND LIMITATIONS

Evading Ianus: An attacker could evade the detection of Ianus during account registration by manipulating features or generating fake features that are used by Ianus. We acknowledge that some of our features can be manipulated or faked with relatively low costs. As we discussed in Section 5.2, an attacker could use new and popular OS version and WeChat version, spoof device ID and WiFi MAC, register during daytime instead of late night, as well as specify a location that is consistent with the IP-based location. However, it may take larger costs for an attacker to manipulate the IP based features, phone number based features, and nickname based features. Using only these harder-to-evade features, Ianus still achieves a Precision of 93.6% and a Recall of 45.1%. Finally, we note that an attacker may evade detection of Ianus by recruiting a large number of benign users to register Sybils, which is known as crowdturfing [24, 25, 31, 45, 46]. However, crowdturfing could be detected by other methods, e.g., [57].

Detection coverage: Ianus obtains a Precision that ranges from 92% to 96% on WeChat. Therefore, Ianus can accurately detect Sybils. However, the Recall achieved by Ianus is around 80% (using all features) and around 45% (using harder-to-evade features). We suspect the reason is that some Sybils are registered isolated or manually without significant synchronization and/or anomaly patterns. To address this limitation, Ianus can be used together with the methods that leverage other types of data, e.g., content, behavior, and/or social graphs, to enhance coverage (we will discuss more details on these methods in Section 2).

Retraining: Sybils' registration patterns may change over time. Therefore, Ianus may require retraining when its accuracy decreases significantly. Specifically, WeChat deploys multiple systems (e.g., Ianus and behavior-based) to detect Sybils and WeChat users can also report Sybils. Based on the results of other detection systems (e.g., behavior-based) and user reports, we can decide whether Ianus requires retraining, e.g., if much more Sybils that are detected by the behavior-based system or reported by users are suddenly not detected by Ianus, then Ianus may require retraining.

Applicability of Ianus to other online social networks: Our study focuses on detecting Sybils in WeChat, which has significant and real-world impact as WeChat is a very popular online social network with over 1 billion monthly active users. Moreover, some of our features–such as the features based on IP, phone number, timestamp, OS, and nickname–are applicable to other online social networks (e.g., Facebook and Twitter), while the features based on device ID, WiFi MAC, and "WeChat version" can be further applied to other online social networks that are designed for mobile devices. It would be an interesting future work to extend Ianus to other online social networks.

Rate limiting registration: Sybils share synchronized registration patterns, e.g., they use the same IP addresses or devices. Therefore, limiting the number of registrations per IP or device within a given period of time (e.g., 1 hour) seems a naive method to prevent Sybils. However, many benign accounts also use the same IP addresses for registration. Therefore, such IP-based rate limiting method will substantially influence benign users. Moreover, an attacker can spoof device IDs to evade device-based rate limiting.

7 CONCLUSION AND FUTURE WORK

In this work, we find that Sybils have both synchronized and abnormal registration patterns using real-world labeled registration datasets from WeChat. Synchronized registration patterns mean that attackers use the IPs with the same prefixes, phone numbers from the same areas, same devices, etc. to register Sybils, while abnormal registration patterns mean that the shared attributes are further abnormal for Sybils. Based on the measurement results,

we design Ianus to detect Sybils. In particular, Ianus extracts synchronization and anomaly features for a pair of accounts, uses the features to build a graph in which only Sybils are densely connected with each other, and detects Sybils via analyzing the structure of the graph. Our empirical evaluations on registration datasets from WeChat show that Ianus can effectively detect Sybils. Moreover, Ianus can detect a large amount of Sybils with a high precision in the wild when being deployed at WeChat. Our study shows that Sybils can be effectively detected using registration data. An interesting future work is to further explore unsupervised Sybil detection methods based on registration data.

ACKNOWLEDGMENTS

We would like to thank our shepherd Gianluca Stringhini, and the anonymous reviewers for their insightful comments. This work was supported in part by the National Key R&D Program of China under Grant No. 2018YFB1800304, the National Natural Science Foundation of China under Grant No. 61572278, 61822207, U1736209, and U1636219, DARPA ASED under Grant No. FA8650-18-2-7882, Center for Long-Term Cybersecurity, and the BNRist Network and Software Security Research Program under Grant No. BNR2019TD01004. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of MOST of China, the National Natural Science Foundation of China, DARPA, the National Science Foundation, and BNRist. Qi Li is the corresponding author of this paper.

REFERENCES

[1] 2018. Jieba. https://github.com/fxsjy/jieba.
[2] Lorenzo Alvisi, Allen Clement, Alessandro Epasto, Silvio Lattanzi, and Alessandro Panconesi. 2013. SoK: The Evolution of Sybil Defense via Social Networks. In *IEEE S & P*.
[3] Fabrıcio Benevenuto, Gabriel Magno, Tiago Rodrigues, and Virgılio Almeida. 2010. Detecting spammers on twitter. In *CEAS*.
[4] 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*.
[5] Vincent D Blondel, JeanLoup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics-Theory and Experiment* 2008, 10 (2008), 155–168.
[6] Yazan Boshmaf, Dionysios Logothetis, Georgos Siganos, Jorge Lería, Jose Lorenzo, Matei Ripeanu, and Konstantin Beznosov. 2015. Integro: Leveraging Victim Prediction for Robust Fake Account Detection in OSNs.. In *NDSS*, Vol. 15. 8–11.
[7] Elie Bursztein, Jonathan Aigrain, Angelika Moscicki, and John C. Mitchell. 2014. The end is nigh: Generic solving of text-based captchas. In *WOOT*.
[8] Elie Bursztein, Romain Beauxis, Hristo Paskov, Daniele Perito, Celine Fabry, and John Mitchell. 2011. The Failure of Noise-Based Non-continuous Audio Captchas. In *IEEE Symposium on Security and Privacy*. 19 – 31.
[9] Elie Bursztein, Matthieu Martin, and John C. Mitchell. 2011. Text-based CAPTCHA Strengths and Weaknesses. In *CCS*. 125–138.
[10] Zhuhua Cai and Christopher Jermaine. 2012. The Latent Community Model for Detecting Sybils in Social Networks. In *NDSS*.
[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*. 477–488.
[13] G. Danezis and P. Mittal. 2009. SybilInfer: Detecting Sybil Nodes using Social Networks. In *NDSS*.
[14] John R. Douceur. 2002. The Sybil Attack. In *IPTPS*.
[15] Matthew Edwards, Guillermo Suarez-Tangil, Claudia Peersman, Gianluca Stringhini, Awais Rashid, and Monica Whitty. 2018. The Geography of Online Dating Fraud. In *ConPro*.
[16] Manuel Egele, Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. 2015. Towards Detecting Compromised Accounts on Social Networks. *IEEE Transactions on Dependable and Secure Computing* 12, 2 (2015), 447–460.
[17] D. Freeman, M. Dürmuth, and B. Biggio. 2016. Who are you? A statistical approach to measuring user authenticity. In *NDSS*.
[18] Hongyu Gao, Jun Hu, Christo Wilson, Zhichun Li, Yan Chen, and Ben Y Zhao. 2010. Detecting and characterizing social spam campaigns. In *IMC*. 35–47.
[19] 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 *2018 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 1–9.
[20] Neil Zhenqiang Gong, Mario Frank, and Prateek Mittal. 2014. Sybilbelief: A semi-supervised learning approach for structure-based sybil detection. *IEEE Transactions on Information Forensics and Security* 9, 6 (2014), 976–987.
[21] Hacking Financial Market. 2016. http://goo.gl/4AkWyt
[22] Shuang Hao, Alex Kantchelian, Brad Miller, Vern Paxson, and Nick Feamster. 2016. PREDATOR: Proactive Recognition and Elimination of Domain Abuse at Time-Of-Registration. In *CCS*.
[23] Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. 2017. Random walk based fake account detection in online social networks. In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 273–284.
[24] Kyumin Lee, Prithivi Tamilarasan, and James Caverlee. 2013. Crowdturfers, Campaigns, and Social Media: Tracking and Revealing Crowdsourced Manipulation of Social Media. In *ICWSM*.
[25] Kyumin Lee, Steve Webb, and Hancheng Ge. 2014. The Dark Side of Micro-Task Marketplaces: Characterizing Fiverr and Automatically Detecting Crowdturfing. *CoRR* abs/1406.0574 (2014).
[26] Anna Leontjeva, Moises Goldszmidt, Yinglian Xie, Fang Yu, and Martín Abadi. 2013. Early security classification of skype users via machine learning. In *AISec*.
[27] Changchang Liu, Peng Gao, Matthew Wright, and Prateek Mittal. 2015. Exploiting temporal dynamics in Sybil defenses. In *CCS*. 805–816.
[28] Abedelaziz Mohaisen, Nicholas Hopper, and Yongdae Kim. 2011. Keep your friends close: Incorporating trust into social network-based Sybil defenses. In *IEEE INFOCOM*.
[29] Abedelaziz Mohaisen, Aaram Yun, and Yongdae Kim. 2010. Measuring the mixing time of social graphs. In *IMC*.
[30] Jonghyuk Song, Sangho Lee, and Jong Kim. 2011. Spam filtering in Twitter using sender-receiver relationship. In *RAID*.
[31] Jonghyuk Song, Sangho Lee, and Jong Kim. 2015. CrowdTarget: Target-based Detection of Crowdturfing in Online Social Networks. In *CCS*. 793–804.
[32] Andreas Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*.
[33] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. 2010. Detecting spammers on social networks. In *ACSAC*.
[34] 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*. 563–578.
[35] 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*.
[36] Kurt Thomas, Danny Yuxing Huang, David Wang, Elie Bursztein, Chris Grier, Thomas J. Holt, Christopher Kruegel, Damon McCoy, Stefan Savage, and Giovanni Vigna. 2015. Framing Dependencies Introduced by Underground Commoditization. In *WEIS*.
[37] Kurt Thomas, Frank Li, Chris Grier, and Vern Paxson. 2014. Consequences of connectivity: Characterizing account hijacking on twitter. In *CCS*. 489–500.
[38] Kurt Thomas, Damon Mccoy, Alek Kolcz, Alek Kolcz, and Vern Paxson. 2013. Trafficking fraudulent accounts: the role of the underground market in Twitter spam and abuse. In *Usenix Security Symposium*. 195–210.
[39] Bimal Viswanath, Ansley Post, Krishna P. Gummadi, and Alan Mislove. 2010. An Analysis of Social Network-Based Sybil Defenses. In *ACM SIGCOMM*.
[40] Alex Hai Wang. 2010. Don't Follow Me - Spam Detection in Twitter. In *SECRYPT 2010*.
[41] 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 *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 465–474.
[42] Binghui Wang, Jinyuan Jia, and Neil Zhenqiang Gong. 2018. Graph-based security and privacy analytics via collective classification with joint weight learning and propagation. *arXiv preprint arXiv:1812.01661* (2018).
[43] Binghui Wang, Le Zhang, and Neil Zhenqiang Gong. 2017. SybilSCAR: Sybil detection in online social networks via local rule based propagation. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 1–9.
[44] Gang Wang, Tristan Konolige, Christo Wilson, Xiao Wang, Haitao Zheng, and Ben Y Zhao. 2013. You are how you click: Clickstream analysis for sybil detection. In *USENIX Security Symposium*. 241–256.
[45] Gang Wang, Tianyi Wang, Haitao Zhang, and Ben Y. Zhao. 2014. Man vs. machine: practical adversarial detection of malicious crowdsourcing workers. In *USENIX Security Symposium*. 239–254.
[46] Gang Wang, Christo Wilson, Xiaohan Zhao, Yibo Zhu, Manish Mohanlal, Haitao Zheng, and Ben Y. Zhao. 2012. Serf and turf: crowdturfing for fun and profit. In *WWW*.

[47] 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 *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2012–2022.

[48] Yinglian Xie, Fang Yu, Qifa Ke, Martín Abadi, Eliot Gillum, Krish Vitaldevaria, Jason Walter, Junxian Huang, and Z. Morley Mao. 2012. Innocent by Association: Early Recognition of Legitimate Users. In *CCS*.

[49] Chao Yang, Robert Harkreader, and Guofei Gu. 2011. Die Free or Live Hard? Empirical Evaluation and New Design for Fighting Evolving Twitter Spammers. In *RAID*.

[50] Chao Yang, Robert Harkreader, Jialong Zhang, Seungwon Shin, and Guofei Gu. 2012. Analyzing Spammer's Social Networks for Fun and Profit. In *WWW*.

[51] Zhi Yang, Jilong Xue, Xiaoyong Yang, Xiao Wang, and Yafei Dai. 2016. VoteTrust: Leveraging Friend Invitation Graph to Defend against Social Network Sybils. *IEEE Transactions on Dependable and Secure Computing* 13, 4 (2016), 488–501.

[52] Guixin Ye, Zhanyong Tang, Dingyi Fang, Zhanxing Zhu, Yansong Feng, Pengfei Xu, Xiaojiang Chen, and Zheng Wang. 2018. Yet another text captcha solver: A generative adversarial network based approach. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*.

[53] Xu ying Liu, Jianxin Wu, Zhi hua Zhou, and Senior Member. 2009. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39, 2 (2009).

[54] 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*.

[55] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. 2006. SybilGuard: Defending Against Sybil Attacks via Social Networks. In *SIGCOMM*.

[56] Yao Zhao, Yinglian Xie, Fang Yu, Qifa Ke, Yuan Yu, Yan Chen, and Eliot Gillum. 2009. BotGraph: Large Scale Spamming Botnet Detection. In *NSDI*.

[57] Haizhong Zheng, Minhui Xue, Hao Lu, Shuang Hao, Haojin Zhu, Xiaohui Liang, and Keith Ross. 2018. Smoke Screener or Straight Shooter: Detecting Elite Sybil Attacks in User-Review Social Networks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

[58] Yang Zhi, Christo Wilson, Tingting Gao, Tingting Gao, Ben Y. Zhao, and Yafei Dai. 2011. Uncovering social network Sybils in the wild. *Acm Transactions on Knowledge Discovery from Data* 8, 1 (2011), 2.