Íntegro: Leveraging Victim Prediction for Robust Fake Account Detection in OSNs

Yazan Boshmaf*, Dionysios Logothetis†, Georgos Siganos‡, Jorge Lería§,
Jose Lorenzo§, Matei Ripeanu*, and Konstantin Beznosov*
*University of British Columbia
†Telefonica Research §Tuenti, Telefonica Digital
‡Qatar Computing Research Institute

***Abstract*—Detecting fake accounts in online social networks (OSNs) protects OSN operators and their users from various malicious activities. Most detection mechanisms attempt to predict and classify user accounts as real (i.e., benign, honest) or fake (i.e., malicious, Sybil) by analyzing user-level activities or graph-level structures. These mechanisms, however, are not robust against adversarial attacks in which fake accounts cloak their operation with patterns resembling real user behavior.**

We herein demonstrate that victims, benign users who control real accounts and have befriended fakes, form a distinct classification category that is useful for designing robust detection mechanisms. First, as attackers have no control over victim accounts and cannot alter their activities, a victim account classifier which relies on user-level activities is relatively harder to circumvent. Second, as fakes are directly connected to victims, a fake account detection mechanism that integrates victim prediction into graph-level structures is more robust against manipulations of the graph.

To validate this new approach, we designed Íntegro, a scalable defense system that helps OSNs detect automated fake accounts using a robust user ranking scheme. Íntegro starts by predicting victim accounts from user-level activities. After that, it integrates these predictions into the graph as weights, so that edges incident to predicted victims have much lower weights than others. Finally, Íntegro ranks user accounts based on a modified random walk that starts from a known real account. Íntegro guarantees that most real accounts rank higher than fakes so that OSN operators can take actions against low-ranking fake accounts.

We implemented Íntegro using widely-used, open-source distributed computing platforms in which it scaled nearly linearly. We evaluated Íntegro against SybilRank, the state-of-the-art in fake account detection, using real-world datasets and a large-scale deployment at Tuenti, the largest OSN in Spain. We show that Íntegro significantly outperforms SybilRank in user ranking quality, where the only requirement is to employ a victim classifier is better than random. Moreover, the deployment of Íntegro at Tuenti resulted in up to an order of magnitude higher precision in fake accounts detection, as compared to SybilRank.

NDSS '15, 8-11 February 2015, San Diego, CA, USA
Copyright 2015 Internet Society, ISBN 1-891562-38-X
http://dx.doi.org/10.14722/ndss.2015.23260

I. Introduction

The rapid growth of online social networks (OSNs), such as Facebook, Twitter, RenRen, LinkedIn, Google+, and Tuenti, has been followed by an increased interest in abusing them. Due to their open nature, OSNs are particularly vulnerable to the *Sybil attack* [1], where an attacker creates multiple fake accounts called *Sybils* for various adversarial objectives.

The problem. In its 2014 earnings report, Facebook estimated that up to 15 millions (1.2%) of its monthly active users are in fact "undesirable," representing fake accounts that are used in violation of the site's terms of service [2]. For such OSNs, the existence of fakes leads advertisers, developers, and investors to distrust their reported user metrics, which negatively impacts their revenues [3]. Attackers create and automate fake accounts for various malicious activities, including social spamming [4], malware distribution [5], political astroturfing [6], and private data collection [7]. It is therefore important for OSNs to detect fake accounts as fast and accurately as possible.

The challenge. Most OSNs employ detection mechanisms that attempt to identify fake accounts through analyzing either user-level activities or graph-level structures. In the first approach, unique *features* are extracted from recent user activities (e.g., frequency of friend requests, fraction of accepted requests), after which they are applied to a classifier that has been trained offline using machine learning techniques [8]. In the second approach, an OSN is formally modeled as a *graph*, with nodes representing user accounts and edges representing social relationships (e.g., friendships). Given the assumption that *fakes can befriend only few real accounts*, the graph is partitioned into two regions separating real accounts from fakes, with a narrow passage between them [9]. While these techniques are effective against naïve attacks, various studies showed they are inaccurate in practice and can be easily evaded [7], [10], [11]. For example, attackers can cheaply create fakes that resemble real users, circumventing feature-based detection, or use simple social engineering tactics to befriend a large number of real users, invalidating the assumption behind graph-based detection. In this work, we aim to tackle the question: "How can we design a *robust* defense mechanism that allows an OSN to detect accounts which are highly likely to be fake?"

Implications. If an OSN operator can detect fakes efficiently and effectively, it can improve the experience of its users by thwarting annoying spam messages and other abusive content. The OSN operator can also increase the credibility of its user metrics and enable third parties to consider its user accounts as authentic digital identities [12]. Moreover, the operator can better utilize the time of its analysts who manually inspect and validate accounts based on user reports. For example, Tuenti, the largest OSN in Spain with 15M active users, estimates that only 5% of the accounts inspected based on user reports are in fact fake, which signifies the inefficiency of this manual process [13]. The OSN operator can also selectively enforce abuse mitigation techniques, such as CAPTCHA challenges [8] and photo-based social authentication [14], to suspicious accounts while running at a lower risk of annoying benign users.

Our solution. We present Íntegro, a robust defense system that helps OSNs identify fake accounts, *which can befriend many real accounts*, through a user ranking scheme.[1] We designed Íntegro for OSNs whose social relationships are bidirectional (e.g., Facebook, Tuenti, LinkedIn), with the ranking process being completely transparent to users. While Íntegro's ranking scheme is graph-based, the social graph is preprocessed first and annotated with information derived from feature-based detection techniques. This approach of integrating user-level activities into graph-level structures positions Íntegro as the first feature-and-graph-based detection mechanism.

Our design is based on the observation that *victim accounts*, real accounts whose users have accepted friend requests sent by fakes, are useful for designing robust fake account detection mechanisms. In particular, Íntegro uses basic account features (e.g., gender, number of friends, time since last update), which are cheap to extract from user-level activities, in order to train a classifier to predict unknown victims in the OSN. As attackers do not have control over victims nor their activities, a victim classifier is inherently more resilient to adversarial attacks than a similarly-trained fake account classifier. Moreover, as victims are directly connected to fakes, they form a "borderline" separating real accounts from fakes in the social graph. Íntegro makes use of this observation by incorporating victim predictions into the graph as weights, such that edges incident to predicted victims have lower weights than others. Finally, Íntegro ranks user accounts based on the landing probability of a modified random walk that starts from a known real account. The walk is "short" by terminating its traversal early before it converges. The walk is "supervised" by biasing its traversal towards nodes that are reachable through higher-weight paths. As this short, supervised random walk is likely to stay within the subgraph consisting of real accounts, most real accounts receive higher ranks than fakes. Unlike SybilRank [13], the state-of-the-art in graph-based fake account detection, *we do not assume sparse connectivity between real and fake accounts*, which makes Íntegro the first fake account detection system that is robust against adverse manipulation of the graph.

For an OSN consisting of n users, Íntegro takes $O(n \log n)$ time to complete its computation. For attackers who randomly establish a set E_a of edges between victim and fake accounts, Íntegro guarantees that no more than $O(\mathrm{vol}(E_a) \log n)$ fakes are assigned ranks similar to or higher than real accounts in the worst case, where $\mathrm{vol}(E_a)$ is the sum of weights on edges in E_a. Even with a random victim classifier that labels accounts as victims with 0.5 probability, Íntegro ensures that $\mathrm{vol}(E_a)$ is at most equals to $|E_a|$, resulting in an improvement factor of $O\left(|E_a|/\mathrm{vol}(E_a)\right)$ over SybilRank.

Main results. We evaluated Íntegro against SybilRank using real-world datasets and a large-scale deployment at Tuenti. We chose SybilRank because it was shown to outperform known contenders [13], including EigenTrust [15], SybilGuard [16], SybilLimit [17], SybilInfer [18], Mislove's method [19], and GateKeeper [20]. In addition, as SybilRank relies on a ranking scheme that is similar to ours, albeit on an unweighted graph, evaluating against SybilRank allowed us to show the impact of leveraging victim prediction on ranking quality. Our results show that Íntegro consistently outperforms SybilRank in user ranking quality, especially as E_a grows large. In particular, Íntegro resulted in up to 30% improvement over SybilRank in the ranking's area under ROC curve (AUC), which represents the probability that a random real account is ranked higher than a random fake account.

In practice, the deployment of Íntegro at Tuenti resulted in up to an order of magnitude higher precision in fake account detection, where ideally fakes should be located at the bottom of the ranked user list. In particular, for the bottom 20K low-ranking users, Íntegro achieved 95% precision, as compared to 43% by SybilRank and 5% by Tuenti's user-based abuse reporting system. More importantly, the precision dramatically decreased when moving up in the ranked list, which means Íntegro consistently placed most of the fakes at the bottom of the list, unlike SybilRank. The only requirement with Íntegro is to use a victim classifier that is better than random. This can be easily achieved during the cross-validation phase by deploying a victim classifier with an AUC greater than 0.5.

From system scalability standpoint, Íntegro scales to OSNs with many million users and runs on commodity machines. We implemented Íntegro on top of open-source implementations of MapReduce [21] and Pregel [22]. Using a synthetic benchmark of an OSN consisting of 160M users, Íntegro takes less than 30 minutes to finish its computation on 33 commodity machines.

Contributions. This work makes the following contributions:

• *Integrating user-level activities into graph-level structures.* We presented the design and analysis of Íntegro, a fake account detection system that relies on a novel technique for integrating user-level activities into graph-level structures. Íntegro uses feature-based detection with user-level activities to predict how likely each user is to be a victim. By weighting the graph such that edges incident to predicted victims have much lower weights than others, Íntegro guarantees that most real accounts are ranked higher than fakes. These ranks are derived from the landing probability of a modified random walk that starts from a known real account. To our knowledge, Íntegro is the first detection system that is robust against adverse manipulation of the social graph, where fakes follow an adversarial strategy to befriend a large number of accounts, real or fake, in an attempt to evade detection (Sections III and IV).

• *Implementation and evaluation.* We implemented Íntegro on top of widely-used, open-source distributed machine learning and graph processing platforms. We evaluated Íntegro against SybilRank using real-world datasets and a large-scale deployment at Tuenti. In practice, Íntegro has allowed Tuenti to detect at least 10 times more fakes than their current user-based abuse reporting system, where reported users are not ranked. With an average of 16K reports per day [13], this improvement has been useful to both Tuenti and its users (Sections V and VI).

[1] In Spanish, the word "íntegro" means integrated, which suites our approach of integrating user-level activities into graph-level structures.

II. Background and Related Work

We first outline the threat model we assume in this work. We then present required background and related work on fake account detection, abuse mitigation and the ground-truth, social infiltration, and analyzing victims of fakes in OSNs.

A. Threat model

We focus on OSNs such as Facebook, RenRen, and Tuenti, which are open to everyone and allow users to declare bilateral relationships (i.e., friendships).

Capabilities. We consider attackers who are capable of creating and automating fake accounts on a large scale [23]. Each fake account, also called a *socialbot* [24], can perform social activities similar to those of real users. This includes sending friend requests and posting social content. We do not consider attackers who are capable of hijacking real accounts, as there are existing detection systems that tackle this threat (e.g., COMPA [25]). We focus on detecting fake accounts that can befriend a large number of benign users in order to mount subsequent attacks, as we describe next.

Objectives. The objective of an attacker is to distribute spam and malware, misinform, or collect private user data on a large scale. To achieve this objective, the attacker has to *infiltrate* the target OSN by using the fakes to befriend many real accounts. Such an infiltration is required because isolated fake accounts cannot directly interact with or promote content to most users in the OSN [23]. This is also evident by a thriving underground market for social infiltration. For example, attackers can now connect their fake accounts with 1K users for $26 or less [26].

Victims. We refer to benign users who have accepted friend requests from fake accounts as *victims*. We refer to friendships between victims and fakes as *attack edges*. Victims control real accounts and engage with others in non-adversarial activities.

B. Fake account detection

From a systems design perspective, most of today's fake account detection mechanisms are either feature-based or graph-based, depending on whether they utilize machine learning or graph analysis techniques in order to identify fakes. Next, we discuss each of these approaches in detail.

Feature-based detection. This approach relies on user-level activities and its account details (i.e., user logs, profiles). By identifying unique features of an account, one can classify each account as fake or real using various machine learning techniques. For example, Facebook employs an "immune system" that performs real-time checks and classification for each read and write action on its database, which are based on features extracted from user accounts and their activities [8].

Yang *et al.* used ground-truth provided by RenRen to train an SVM classifier in order to detect fake accounts [27]. Using simple features, such as frequency of friend requests, fraction of accepted requests, and per-account clustering coefficient, the authors were able to train a classifier with 99% true-positive rate (TPR) and 0.7% false-positive rate (FPR).

Stringhini *et al.* utilized honeypot accounts to collect data describing various user activities in OSNs [28]. By analyzing the collected data, they were able to build a ground-truth for real and fake accounts, with features similar to those outlined above. The authors trained two random forests (RF) classifiers to detect fakes in Facebook and Twitter, ending up with 2% FPR and 1% false-negative rate (FNR) for Facebook, and 2.5% FPR and 3% FNR for Twitter.

Wang *et al.* used a click-stream dataset provided by RenRen to cluster user accounts into "similar" behavioral groups, corresponding to real or fake accounts [29]. Using the METIS clustering algorithm [30] with both session and clicks features, such as average clicks per session, average session length, the percentage of clicks used to send friend requests, visit photos, and share content, the authors were able to calibrate a cluster-based classifier with 3% FPR and 1% FNR.

Even though feature-based detection scales to large OSNs, it is still relatively easy to circumvent. This is the case because it depends on features describing activities of known fakes in order to identify unknown ones. In other words, attackers can evade detection by adversely modifying the content and activity patterns of their fakes, leading to an arms race [31]–[33]. Also, feature-based detection does not provide any formal security guarantees and often results in a high FPR in practice. This is partly attributed to the large variety and unpredictability of behaviors of users in adversarial settings [13].

With Íntegro, we employ feature-based detection to identify unknown victims in a *non-adversarial* setting. The dataset used to train a victim classifier includes features of only known real accounts that have either accepted or rejected friend requests send by known fakes. As real accounts are controlled by benign users who are not adversarial, a feature-based victim account classifier is harder to circumvent than a similarly-trained fake account classifier. As we discuss in Section IV, we only require victim classification to be better than random guessing in order to outperform the state-of-the-art in fake account detection.

Graph-based detection. As a response to the lack of formal security guarantees in feature-based detection, the state-of-the-art in fake account detection utilizes a graph-based approach instead. In this approach, an OSN is modeled as a graph, with nodes representing user accounts and edges between nodes representing social relationship. Given the assumption that fakes can establish only a small number of attack edges, the subgraph induced by the set of real accounts is sparsely connected to fakes, that is, the *cut* which crosses over attack edges is sparse.[2] Graph-based detection mechanisms make this assumption, and attempt to find such a sparse cut with formal guarantees [34]–[36]. For example, Tuenti employs SybilRank to rank accounts according to their perceived likelihood of being fake, based on structural properties of its social graph [13].

Yu *et al.* were among the first to analyze the social graph for the purpose of identifying fake accounts in OSNs [16], [17]. The authors developed a technique that labels each account as either fake or real based on multiple, modified random walks. This binary classification is used to partition the graph into two smaller subgraphs that are sparsely interconnected via attack edges, separating real accounts from fakes. They also proved that in the worst case $O(|E_a| \log n)$ fakes can be misclassified, where $|E_a|$ is the number of attack edges and n is the number

[2] A cut is a partition of nodes into two disjoint subsets. Visually, it is a line that cuts through or crosses over a set of edges in the graph (see Fig. 2).

of accounts in the network. Accordingly, it is sufficient for the attacker to establish $\Omega(n/\log n)$ attack edges in order to evade this detection scheme with 0% TPR.

Viswanath *et al.* employed community detection techniques to identify fake accounts in OSNs [19]. In general, community detection decomposes a given graph into a number of tightly-knit subgraphs that are loosely connected to each other, where each subgraph is called a *community* [37], [38]. By expanding a community starting with known real accounts [39], the authors were able to identify the subgraph which contains mostly real accounts. Recently, however, Alvisi *et al.* showed that this local community detection technique can be easily circumvented if the fakes establish sparse connectivity among themselves [9].

As binary classification often leads to high FPR [19], Cao *et al.* proposed to rank the users instead so that most fakes are ranked lower than real accounts [13]. The authors developed SybilRank, a fake account detection system that assigns each account a rank describing how likely it is to be fake based on a modified random walk, where a lower rank means the account is more likely to be fake. They also proved that $O(|E_a|\log n)$ fakes can outrank real accounts in the worst case, given the fakes establish $|E_a|$ attack edges with victims at random.

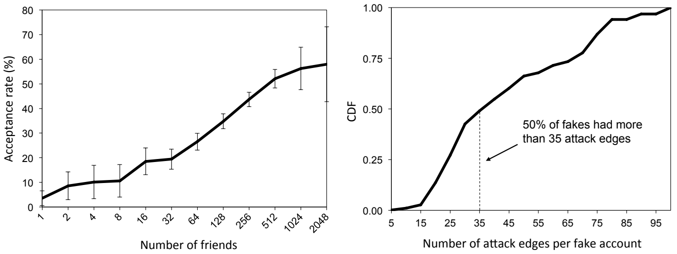
While graph-based detection offers desirable security guarantees, real-world social graphs do not conform with the main assumption on which it depends. In particular, various studies confirmed that attackers can infiltrate OSNs on a large scale by deceiving users into befriending their fakes [7], [10], [11]. As we discuss next, social infiltration renders graph-based fake account detection ineffective in practice.

With Íntegro, we *do not assume that fakes are limited by how many attack edges they can establish.* We instead leverage victim prediction to weight the graph and bound the security guarantee by the aggregate weight on attack edges, $\mathrm{vol}(E_a)$, rather than their number, $|E_a|$. In particular, by assigning lower weights to edges incident to potential victims, we upper bound the value of $\mathrm{vol}(E_a)$ by $|E_a|$, as we show later in Section IV.

C. Abuse mitigation and the ground-truth

Due to the inapplicability of automated account suspension, OSNs employ abuse mitigation techniques, such as CAPTCHA challenges [8] and photo-based social authentication [14], so as to rate-limit accounts that have been automatically flagged as fake or suspicious. Moreover, these accounts are pooled for manual inspection by experienced analysts who build a ground-truth of real and fake accounts along with their features, before suspending or removing verified fakes [8], [13], [27], [40].

While maintaining an up-to-date ground-truth is important for retraining deployed classifiers and estimating how effective they are in practice, it is rather a time-consuming and non-scalable task. For example, on an average day, each analyst at Tuenti inspects 250–350 accounts an hour, and for a team of 14 employees, up to 30K accounts are inspected per day [13]. It is thus important to rank user accounts in terms of how likely they are to be fake in order to prioritize account inspection by analysts. Íntegro offers this functionality and leads to a faster reaction against potential abuse by fakes, benefiting both OSN operators and their users.



(a) User Susceptibility **(b)** Attack scalability

Fig. 1: Social infiltration in Facebook. In (a), while the fakes did not share mutual friends with invited users, the more friends these users had the more likely it was for them to accept friend requests sent by the fakes (CI=95%). In (b), contrary to what is often assumed in literature, fake accounts can use simple automated social engineering to establish a large number of attack edges.

D. Social infiltration

In early 2011, we conducted a study to evaluate how easy it is to infiltrate large OSNs such as Facebook [23]. In particular, we used 100 automated fake accounts to send friend requests to 9.6K real users, where each user received exactly one request.

Main results. We found that users are not careful in their befriending decisions, especially when they share mutual friends with the requester. This behavior was exploited by the fakes to achieve large-scale social infiltration with a success rate of up to 80%, in which case the fakes shared at least 11 mutual friends with the victims. In particular, we reported two main results that are important for designing fake account detection systems. First of all, some users are more likely to be victims than others. As shown in Fig. 1a, the more friends a user has, the more likely the user is to accept friend requests sent by fakes posing as strangers, regardless of their gender or number of mutual friends. Second, attack edges are generally easy to establish in OSN such as Facebook. As shown in Fig. 1b, an attacker can establish enough attack edges such that there is no sparse cut separating real accounts from fakes [36].

Implications. The study suggests that one can predict victims of fake accounts from user-level activities using low-cost features (e.g., number of friends). In addition, the study shows that graph-based detection mechanisms that rely solely on the graph structure are not effective under social infiltration. As social infiltration is prominent in other OSNs [41], [42], new proposals for graph-based detection should extend their threat model and include attackers who can infiltrate on a large scale.

E. Analyzing victim accounts

While we are the first to utilize victim accounts to separate fakes from real accounts, others have analyzed victim accounts as part of the larger cyber criminal ecosystem in OSNs [43].

Wagner *et al.* developed predictive models to identify users who are more susceptible to social infiltration in Twitter [11]. They found that susceptible users, also called *potential victims*, tend to use Twitter for conversational purposes, are more open and social since they communicate with many different users, use more socially welcoming words, and show higher affection than non-susceptible users.

Yang *el al.* studied the cyber criminal ecosystem on Twitter [44]. They found that victims fall into one of three categories. The first are *social butterflies* who have large numbers of followers and followings, and establish social relationships with other accounts without careful examination. The second are *social promoters* who have large following-follower ratios, larger following numbers, and a relatively high URL ratios in their tweets. These victims use Twitter to promote themselves or their business by actively following other accounts without consideration. The last are *dummies* who post few tweets but have many followers. These victims are actually dormant fake accounts at an early stage of their abuse.

III. Intuition, Goals, and Model

We now introduce Íntegro, a fake account detection system that is *robust against social infiltration*. We first present the intuition behind our design, followed by its goals and model.

A. Intuition

Some users are more likely to become victims than others. If we can train a classifier to accurately predict whether a user is a victim with some probability, we can then identify the cut which separates fakes from real accounts in the graph. As victims are benign users who are not adversarial, the output of this classifier represents a reliable information which we can integrate in the graph. To find the cut which crosses over mostly attack edges, we can define a graph weighting scheme that assigns edges incident to predicted victims lower weights than others, where weight values are calculated from prediction probabilities. In a weighted graph, the sparsest cut is the cut with the smallest *volume*, which is the sum of weights on edges across the cut. Given an accurate victim classifier, such a cut is expected to cross over some or all attack edges, effectively separating real accounts from fakes, even if the number of attack edges is large. We find this cut using a ranking scheme that ideally assigns higher ranks to nodes in one partition of the cut than others. This ranking scheme is inspired by similar graph partitioning algorithms proposed by Spielman *et al.* [45], Yu [34], and Cao *et al.* [13].

B. Design goals

Íntegro aims to help OSN operators in detecting fake accounts using a meaningful user ranking scheme. In particular, Íntegro has the following design goals:

• *High-quality user ranking (effectiveness).* The system should consistently assign higher ranks to real accounts than fakes. It should limit the number of fakes that might rank similar to or higher than real accounts. The system should be robust against social infiltration under real-world attack strategies. Given a ranked list of users, a high percentage of the users at the bottom of the list should be fake. This percentage should decrease as we go up in the list.

• *Scalability (efficiency).* The system should have a practical computational cost which allows it to scale to large OSNs. It should deliver ranking results in only few minutes. The system should be able to extract useful, low-cost features and process large graphs on commodity machines, in order to allow OSNs to deploy it on their existing computer clusters.

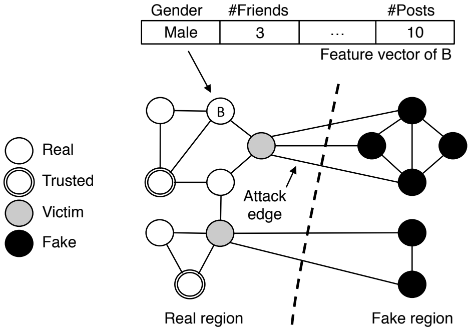


Fig. 2: System model. In this figure, the OSN is represented as a graph consisting of 14 users. There are 8 real accounts, 6 fake accounts, and 5 attack edges. The cut, represented by a dashed-line, partitions the graph into two regions, real and fake. Victim accounts are real accounts that are directly connected to fakes. Trusted accounts are accounts that are known to be real and not victims. Each account has a feature vector representing basic account information. Initially, all edges have a unit weight, so user B for example has a degree of 3.

C. System model

As illustrated in Fig. 2, we model an OSN as an undirected graph $G = (V, E)$, where each node $v_i \in V$ represents a user account and each edge $\{v_i, v_j\} \in E$ represents a bilateral social relationship among v_i and v_j. In the graph G, there are $n = |V|$ nodes and $m = |E|$ edges.

Attributes. Each node $v_i \in V$ has a *degree* $\deg(v_i)$ that is equal to the sum of weights on edges incident to v_i. Moreover, v_i has a *feature vector* $\mathcal{A}(v_i)$, where each entry $a_j \in \mathcal{A}(v_i)$ describes a feature or an attribute of the account v_i. Each edge $\{v_i, v_j\} \in E$ has a *weight* $w(v_i, v_j) \in (0, 1]$, which is initially set to $w(v_i, v_j) = 1$.

Regions. The node set V is divided into two disjoint sets, V_r and V_f, representing real and fake accounts, respectively. We refer to the subgraph induced by V_r as the *real region* G_r, which includes all real accounts and the friendships between them. Likewise, we refer to the subgraph induced by V_f as the *fake region* G_f. The regions are connected by a set of attack edges E_a between victim and fake accounts. We assume the OSN operator is aware of a small set of *trusted accounts* V_t, which are known to be real accounts and are not victims.

IV. System Design

We now describe the design behind Íntegro. We start with a short overview of our approach, after which we proceed with a detailed description of each system component.

A. Overview

Íntegro extracts low-cost features from user-level activities in order to train a classifier to identify unknown victims in the social graph. We refer to these accounts as *potential victims*, as there are probabilities attached to their labels. Íntegro then calculates new edge weights from prediction probabilities such that edges incident to identified victims have lower weights than others. Finally, Íntegro ranks user accounts based on the landing probability of a modified random walk that starts from a trusted account picked at random. The walk is "short," as it is terminated early before it converges. The walk is "supervised," as it is biased towards traversing nodes which are reachable via higher-weight paths. This short, supervised random walk has a higher probability to stay in the real region of the graph, as it is highly unlikely to escape into the fake region in few steps through low-weight attack edges. Accordingly, Íntegro assigns most of the real accounts a higher rank than fakes.

B. Identifying potential victims

For each user v_i, Íntegro extracts a feature vector $\mathcal{A}(v_i)$ from its recent user-level activities. A subset of feature vectors is selected to train a binary classifier to predict whether each user is a victim and with what probability. As attackers have no control over victims, such a victim classifier is inherently more resilient to adversarial attacks than similarly-trained fake account classifier. Let us consider one concrete example. In the "boiling-frog" attack [31], fake accounts can force a classifier to tolerate abusive activities by slowly introducing similar activities to the OSN. Because the OSN operator has to retrain deployed classifiers in order to capture new behaviors, a fake account classifier will learn to tolerate more and more abusive activities, until the attacker can launch a full-scale attack without detection [7]. For victim prediction, on the other hand, this is possible only if the accounts used for training have been hijacked. This situation can be avoided by manually verifying the accounts, as described in Section II-C.

Feature engineering. Extracting and selecting useful features from user activities can be both challenging and time consuming. For efficiency, we seek features that can be extracted in $O(1)$ time per user. One candidate location for low-cost feature extraction is the profile page of user accounts, where features are readily available (e.g., a Facebook profile page). However, these features are expected to be statistically "weak," which means they may not strongly correlate with whether a user is a victim or not (i.e., the label). As we explain later, we require the victim classifier to be better than random in order to deliver robust fake account detection. This requirement, fortunately, is easy to satisfy. In particular, we show in Section V that an OSN operator can train and cross-validate a victim classifier that is up to 52% better than random, using strictly low-cost features.

Supervised learning. For each user v_i, Íntegro computes a *vulnerability score* $p(v_i) \in (0,1)$ that represents the probability of v_i to be a victim. For a fixed *operating threshold* $\alpha \in (0,1)$ with a default value of $\alpha = 0.5$, we say v_i is a *potential victim* if $p(v_i) \geq \alpha$. To compute vulnerability scores, Íntegro uses random forests (RF) learning algorithm [46] to train a victim classifier, which given $\mathcal{A}(v_i)$ and α, decides whether the user v_i is a victim with a score $p(v_i)$. We picked this learning algorithm because it is both efficient and robust against model over-fitting [47]. It takes $O(n \log n)$ time to extract n low-cost feature vectors, each consisting of $O(1)$ features, and train a victim classifier. It also takes $O(n)$ to evaluate node scores, given the trained classifier and users' feature vectors.

C. Integrating victim predictions and ranking users

To rank users, Íntegro computes the probability of a modified random walk to land on each user v_i after k steps, where the walk starts from a trusted user account picked at random. For simplicity, we refer to the probability of a random walk to land on a node as its *trust value*, so the probability distribution of the walk at each step can be modeled as a *trust propagation process* [48]. In this process, a weight $w(v_i, v_j)$ represents the rate at which trust may propagate from either side of the edge $\{v_i, v_j\} \in E$. We next describe this process in detail.

Trust propagation. Íntegro utilizes the *power iteration method* to efficiently compute trust values [49]. This method involves successive matrix multiplications where each element of the matrix is the transition probability of the random walk from one node to another. Each iteration computes the trust distribution over nodes as the random walk proceeds by one step. Let $T_k(v_i)$ denote the trust collected by each node $v_i \in V$ after k iterations. Initially, the *total trust*, denoted by $\tau \geq 1$, is evenly distributed among the trusted nodes in V_t:

$$T_0(v_i) = \begin{cases} \tau/|V_t| & \text{if } v_i \in V_t, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The process then proceeds as follows:

$$T_k(v_i) = \sum_{\{v_i,v_j\} \in E} T_{k-1}(v_j) \cdot \frac{w(v_i, v_j)}{\deg(v_j)}, \tag{2}$$

where in iteration k, each node v_i propagates its trust $T_{k-1}(v_i)$ from iteration $k-1$ to each neighbour v_j, proportionally to the ratio $w(v_i, v_j)/\deg(v_i)$. This is required so that the sum of the propagated trust equals $T_{k-1}(v_i)$. The node v_i then collects the trust propagated similarly from each neighbour v_j and updates its trust $T_k(v_i)$. Throughout this process, τ is preserved such that for each iteration $k \geq 1$ we have:

$$\sum_{v_i \in V} T_{k-1}(v_i) = \sum_{v_i \in V} T_k(v_i) = \tau. \tag{3}$$

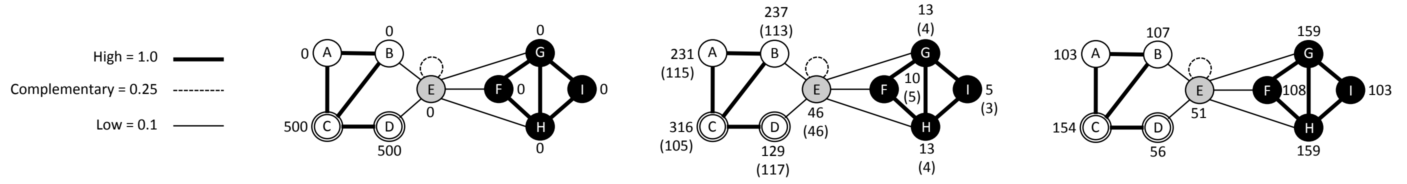
Our goal is to ensure that most real accounts collect higher trust than fake accounts. That is, we seek to limit the portion of τ that escapes the real region G_r and enters the fake region G_f. To achieve this property, we make the following modifications.

Adjusted propagation rates. In each iteration k, the aggregate rate at which τ may enter G_f is strictly limited by the sum of weights on the attack edges, which we denote by the *volume* $\text{vol}(E_a)$. Therefore, we aim to adjust the weights in the graph such that $\text{vol}(E_a) \in (0, |E_a|]$, without severely restricting trust propagation in G_r. We accomplish this by assigning smaller weights to edges incident to potential victims than other edges. In particular, each edge $\{v_i, v_j\} \in E$ keeps the default weight $w(v_i, v_j) = 1$ if v_i and v_j are not potential victims. Otherwise, we modify the weight as follows:

$$w(v_i, v_j) = \min\{1, \beta \cdot (1 - \max\{p(v_i), p(v_j)\})\}, \tag{4}$$

where β is a scaling parameter with a default value of $\beta = 2$. Now, as $\text{vol}(E_a) \to 0$ the portion of τ that enters G_f reaches zero as desired. For proper degree normalization, we introduce a self-loop $\{v_i, v_i\}$ with weight $w(v_i, v_i) = (1 - \deg(v_i))/2$ whenever $\deg(v_i) < 1$. Notice that self-loops are considered twice in degree calculation.

Early-terminated propagation. In each iteration k, the *trust vector* $T_k(V) = \langle T_k(v_1), \ldots, T_k(v_n) \rangle$ describes the distribution of τ throughout the graph. As $k \to \infty$ the vector converges



(a) Initialization

(b) After 4 iterations

(c) Stationary distribution

Fig. 3: Trust propagation in a toy graph. Each value is rounded to its nearest natural number. Values in parentheses represent degree-normalized trust (i.e., rank values). In this example, we set $\alpha = 0.5$, $\beta = 2$, $\tau = 1,000$, $p(\cdot) = 0.05$ except for $p(E) = 0.95$, and $\omega = \lceil \log_2(9) \rceil = 4$.

to a stationary distribution $T_\infty(V)$, as follows [50]:

$$T_\infty(V) = \left\langle \tau \cdot \frac{\deg(v_1)}{\mathrm{vol}(V)}, \ldots, \tau \cdot \frac{\deg(v_n)}{\mathrm{vol}(V)} \right\rangle, \tag{5}$$

where the volume $\mathrm{vol}(V)$ in this case is the sum of degrees of nodes in V.[3] In particular, $T_k(V)$ converges after k reaches the *mixing time* of the graph, which is much larger than $O(\log n)$ iterations for various kinds of social networks [37], [51], [52]. Accordingly, we terminate the propagation process early before it converges after $\omega = O(\log n)$ iterations.

Degree-normalization. As described in Equation 5, trust propagation is influenced by individual node degrees. As k grows large, the propagation starts to bias towards high degree nodes. This implies that high degree fake accounts may collect more trust than low degree real accounts, which is undesirable for effective user ranking. To eliminate this node degree bias, we normalize the trust collected by each node by its degree. That is, we assign each node $v_i \in V$ after $\omega = O(\log n)$ iterations a *rank value* $T'_\omega(v_i)$ that is equal to its degree-normalized trust:

$$T'_\omega(v_i) = T_\omega(v_i)/\deg(v_i). \tag{6}$$

Finally, we sort the nodes by their ranks in a descending order.

Example. Fig. 3 depicts trust propagation on a toy graph. In this example, we assume each account has a vulnerability score of 0.05 except the victim E, which has a score of $p(E) = 0.95$. The graph is weighted using $\alpha = 0.5$ and $\beta = 2$, and a total trust $\tau = 1000$ in initialized over the trusted nodes $\{C, D\}$.

After $\omega = 4$ iterations, all real accounts $\{A, B, C, D, E\}$ collect more trust than fake accounts $\{F, G, H, I\}$. The nodes also receive the correct ranking of $(D, A, B, C, E, F, G, H, I)$, as sorted by their degree-normalized trust. In particular, all real accounts have higher rank values than fakes, where the smallest difference is $T'_4(E) - T'_4(F) > 40$. Moreover, notice that real accounts that are not victims have similar rank values, where the largest difference is $T'_4(D) - T'_4(C) < 12$. These sorted rank values, in fact, could be visualized as a stretched-out step function that has a significant drop near the victim's rank value. However, if we allow the process to converge after $k > 50$ iterations, the fakes collect similar or higher trust than real accounts, following Equation 5. Also, notice that the attack edges $E_a = \{\{E, G\}, \{E, F\}, \{E, H\}\}$ have a volume of $\mathrm{vol}(E_a) = 0.3$, which is 10 times lower than its value if the graph had unit weights, with $\mathrm{vol}(E_a) = 3$. As we soon show in Section V, adjusting the propagation rates is essential for robustness against social infiltration.

D. Trusted accounts and community structures

Íntegro is robust against social infiltration, as it limits the portion of τ that enters G_f by the rate $\mathrm{vol}(E_a)$, regardless of the number of attack edges, $|E_a|$. For the case when there are few attack edges so that G_r and G_f are sparsely connected, $\mathrm{vol}(E_a)$ is already small, even if one keeps $w(v_i, v_j) = 1$ for each attack edge $\{v_i, v_j\} \in E_a$. However, G_r is likely to contain communities [37], [53], where each represents a dense subgraph that is sparsely connected to the rest of the graph. In this case, the propagation of τ in G_r becomes restricted by the sparse inter-community connectivity, especially if V_t is contained exclusively in a single community. We therefore seek a selection strategy for trusted accounts, or *seeds*, that takes into account the existing community structure in the graph.

Selection strategy. We pick trusted accounts as follows. First, before rate adjustment, we estimate the community structure in the graph using a community detection algorithm called the *Louvain method* [54]. Second, after rate adjustment, we exclude potential victims and pick small samples of nodes from each detected community at random. Third and last, we inspect the sampled nodes in order to verify they correspond to real accounts that are not victims. We initialize the trust only between the accounts that pass manual verification by experts.

In addition to coping with the existing community structure in the graph, this selection strategy is designed to also reduce the negative impact of *seed-targeting* attacks. In such attacks, fakes befriend trusted accounts in order to adversely improve their ranking, as the total trust τ is initially distributed among trusted accounts. By choosing the seeds at random, however, the attacker is forced to guess the seeds among a large number of nodes. Moreover, by choosing multiple seeds, the chance of correctly guessing the seeds is further reduced, while the amount of trust assigned to each seed in lowered. In practice, the number of seeds depends on available resources for manual account verification, with a minimum of one seed per detected community.

Community detection. We picked the Louvain method as it is both efficient and produces high-quality partitions. The method iteratively groups closely connected communities together to greedily improve the *modularity* of the partition [55], which is a measure for partition quality. In each iteration, every node represents one community, and well-connected neighbors are greedily combined into the same community. At the end of the iteration, the graph is reconstructed by converting the resulting communities into nodes and adding edges that are weighted by inter-community connectivity. Each iteration takes $O(m)$ time, and only a small number of iterations is required to find the

[3] The definition of $\mathrm{vol}(U)$ depends on whether U contains edges or nodes.

community structure which greedily maximizes the modularity.

While one can apply community detection to identify fake accounts [19], doing so hinges on the assumption that fakes always form tightly-knit communities, which is not necessarily true [27]. This also means fakes can easily evade detection if they establish sparse connectivity among themselves [9]. With Íntegro, we do not make such an assumption. In particular, we consider an attacker *who can befriend a large number of real or fake accounts*, without any formal restrictions.

E. Computational cost

For an OSN with n users and m friendships, Íntegro takes $O(n \log n)$ time to complete its computation, end-to-end. We next analyze the running time in detail.

Runtime analysis. Recall that users have a limit on how many friends they can have (e.g., 5K in Facebook, 1K in Tuenti), so we have $O(m) = O(n)$. Identifying potential victims takes $O(n \log n)$ time, where it takes $O(n \log n)$ time to train an RF classifier and $O(n)$ time to compute vulnerability scores. Also, weighting the graph takes $O(m)$ time. Detecting communities takes $O(n)$ time, where each iteration of the Louvain method takes $O(m)$ time, and the graph rapidly shrinks in $O(1)$ time. Propagating trust takes $O(n \log n)$ time, as each iteration takes $O(m)$ time and the propagation process iterates for $O(\log n)$ times. Ranking and sorting users by their degree-normalized trust takes $O(n \log n)$ time. So, the running time is $O(n \log n)$.

F. Security guarantees

For the upcoming security analysis, we consider attackers who establish attack edges with victims uniformly at random. Even though our design does not depend on the actual mixing time of the graph, we assume the real region is *fast mixing* for analytical tractability. This means that it takes $O(\log |V_r|)$ iterations for trust propagation to converge in the real region. In other words, we assume there is a gap between the mixing time of the whole graph and that of the real region such that, after $O(\log n)$ iterations, the propagation reaches its stationary distribution in the real region but not in the whole graph.

Main theoretical result. The main security guarantee provided by Íntegro is captured by the following theoretical result. For a complete proof, we refer the reader to our technical report [56]:

Theorem 4.1: Given a social graph with a fast mixing real region and an attacker who randomly establishes attack edges, the number of fake accounts that rank similar to or higher than real accounts after $O(\log n)$ iterations is $O(\mathrm{vol}(E_a) \log n)$.

Proof sketch: Let us consider a graph $G = (V, E)$ with a fast mixing real region G_r. As weighting a graph changes its mixing time by a constant factor [57], G_r remains fast mixing after rate adjustment.

After $O(\log n)$ iterations, the trust vector $T_\omega(V)$ does not reach its stationary distribution $T_\infty(V)$. Since trust propagation starts from G_r, the fake region G_f gets only a fraction $f < 1$ of the aggregate trust it should receive in $T_\infty(V)$. On the other hand, as the trust τ is conserved during the propagation process (Equation 3), G_r gets $c > 1$ times higher aggregate trust than it should receive in $T_\infty(V)$.

As G_r is fast mixing, each real account $v_i \in V_r$ receives approximately identical rank value of $T'_\omega(v_i) = c \cdot \tau/\mathrm{vol}(V)$, where $\tau/\mathrm{vol}(V)$ is the degree-normalized trust value in $T_\infty(V)$ (Equations 5 and 6). Knowing that G_f is controlled by the attacker, each fake $v_j \in V_f$ receives a rank value $T'_\omega(v_j)$ that depends on how the fakes inter-connect to each other. However, since the aggregate trust in G_f is bounded, each fake receives on average a rank value of $T'_\omega(v_j) = f \cdot \tau/\mathrm{vol}(V)$, which is less than that of a real account. In the worst case, an attacker can arrange a set $V_m \subset V_f$ of fake accounts in G_f such that each $v_k \in V_m$ receives a rank value of $T'_\omega(v_k) = c \cdot \tau/\mathrm{vol}(V)$, while the remaining fakes receive a rank value of zero. Such a set cannot have more than $(f/c) \cdot \mathrm{vol}(V_s) = O(\mathrm{vol}(E_a) \log n)$ accounts, as otherwise, f would not be less than 1 and G_f would receive more than it should in $T_\omega(V)$. ■

Improvement over SybilRank's bound. Íntegro shares many design traits with SybilRank, which is the state-of-the-art in graph-based detection [13]. In particular, modifying Íntegro by setting $w(v_i, v_j) = 1$ for each $(v_i, v_j) \in E$ will in fact result in an identical ranking. It is indeed the prediction and incorporation of potential victims that differentiates Íntegro from other proposals, giving it the unique advantages outlined earlier.

As stated by Theorem 4.1, the bound on ranking quality relies on $\mathrm{vol}(E_a)$, regardless of how large the set E_a grows. As we weight the graph based on the output of the victim classifier, our bound is sensitive to its classification performance. We next prove that if an OSN operator uses a victim classifier that is *uniformly random*, which means each user account $v_i \in V$ is equally vulnerable with $p(v_i) = 0.5$, then Íntegro is as good as SybilRank in terms of ranking quality [13]:

Corollary 4.2: For a uniformly random victims classifier, the number of fake accounts that rank similar to or higher than real accounts after $O(\log n)$ iterations is $O(|E_a| \log n)$.

Proof: This classifier assigns each user account $v_i \in V$ a score $p(v_i) = 0.5$. By Equation 4, each edge $\{v_i, v_j\} \in E$ is assigned a unit weight $w(v_i, v_j) = 1$, where $\alpha = 0.5$ and $\beta = 2$. By Theorem 4.1, the number of fake accounts that rank similar to or higher than real accounts after $\omega = O(\log n)$ iterations is $O(\mathrm{vol}(E_a) \log n) = O(|E_a| \log n)$. ■

By Corollary 4.2, Íntegro can outperform SybilRank in its ranking quality by a factor of $O(|E_a|/\mathrm{vol}(E_a))$, given the used victim classifier is better than random. This can be achieved during the cross-validation phase of the victim classifier, which we thoroughly describe in what follows.

V. System Evaluation

We analyzed and evaluated Íntegro against SybilRank using two real-world datasets recently collected from Facebook and Tuenti. We also compared both systems through a large-scale deployment at Tuenti in collaboration with its "Site Integrity" team, which has 14 full-time account analysts and 10 full-time software engineers who fight spam and other forms of abuse.

Compared system. We chose SybilRank for two main reasons. First, as discussed in Section IV-F, SybilRank utilizes a similar ranking scheme based on the power iteration method, albeit on an unweighted version of the graph. This similarity allowed us to clearly show the impact of leveraging victim prediction on fake account detection. Second, SybilRank outperforms other

Feature	Brief description	Type	RI Score (%)	
			Facebook	Tuenti
User activity:				
Friends	Number of friends the user had	Numeric	100.0	84.5
Photos	Number of photos the user shared	Numeric	93.7	57.4
Feed	Number of news feed items the user had	Numeric	70.6	60.8
Groups	Number of groups the user was member of	Numeric	41.8	N/A
Likes	Number of likes the users made	Numeric	30.6	N/A
Games	Number of games the user played	Numeric	20.1	N/A
Movies	Number of movies the user watched	Numeric	16.2	N/A
Music	Number of albums or songs the user listened to	Numeric	15.5	N/A
TV	Number of TV shows the user watched	Numeric	14.2	N/A
Books	Number of books the user read	Numeric	7.5	N/A
Personal messaging:				
Sent	Number of messages sent by the user	Numeric	N/A	53.3
Inbox	Number of messages in the user's inbox	Numeric	N/A	52.9
Privacy	Privacy level for receiving messages	5-Categorical	N/A	9.6
Blocking actions:				
Users	Number of users blocked by the user	Numeric	N/A	23.9
Graphics	Number of graphics (photos) blocked by the user	Numeric	N/A	19.7
Account information:				
Last updated	Number of days since the user updated the profile	Numeric	90.77	32.5
Highlights	Number of years highlighted in the user's time-line	Numeric	36.3	N/A
Membership	Number of days since the user joined the OSN	Numeric	31.7	100
Gender	User is male or female	2-Categorical	13.8	7.9
Cover picture	User has a cover picture	2-Categorical	10.5	< 0.1
Profile picture	User has a profile picture	2-Categorical	4.3	< 0.1
Pre-highlights	Number of years highlighted before 2004	Numeric	3.9	N/A
Platform	User disabled third-party API integration	2-Categorical	1.6	< 0.1

TABLE I: Low-cost features extracted from Facebook and Tuenti datasets. The RI score is the relative importance of the feature. A value of "N/A" means the feature was not available for this dataset. A k-Categorical feature means this feature can have one value out of k categories (e.g., boolean features are 2-Categorical).

contenders [13], including EigenTrust [15], SybilGuard [16], SybilLimit [17], SybilInfer [18], Mislove's method [19], and GateKeeper [20]. We next contrast these systems to both SybilRank and Íntegro.

SybilGuard [16] and SybilLimit [17] identify fake accounts based on a large number of modified random walks, where the computational cost is $O(\sqrt{m}n\log n)$ in centralized setting like OSNs. SybilInfer [18], on the other hand, uses Bayesian inference techniques to assign each user account a probability of being fake in $O(n(\log n)^2)$ time per trusted account. The system, however, does not provide analytical bounds on how many fakes can outrank real accounts in the worst case.

GateKeeper [20], which is a flow-based detection approach, improves over SumUp [58]. It relies on strong assumptions that require balanced graphs and costs $O(n\log n)$ time per trusted account, referred to as a "ticket source."

Viswanath *et al.* used Mislove's algorithm [39] to greedily expand a local community around known real accounts in oder to partition the graph into two communities representing real and fake regions [19]. This algorithm, however, costs $O(n^2)$ time and its detection can be easily evaded if the fakes establish sparse connectivity among themselves [9].

Compared to these systems, SybilRank provides an equivalent or tighter security bound and is more computationally efficient, as it requires $O(n\log n)$ time regardless of the number of trusted accounts. Compared to SybilRank, Íntegro provides $O(|E_a|/\mathrm{vol}(E_a))$ improvement on its security bound, requires the same $O(n\log n)$ time, and is robust against social infiltration, unlike SybilRank and all other systems.

A. Datasets

We used two datasets from two different OSNs. The first dataset was collected in the study described in Section II-D, and contained public user profiles and two graph samples. The second dataset was collected from Tuenti's production servers, and contained a day's worth of server-cached user profiles.

Research ethics. For collecting the first dataset, we followed known practices and obtained the approval of our university's research ethics board [7]. As for the second dataset, we signed a non-disclosure agreement with Tuenti in order to access an anonymized, aggregated version of its user data, with the whole process being mediated by Tuenti's Site Integrity team.

The ground-truth. For the Tuenti dataset, the accounts were inspected and labeled by its accounts' analysts. The inspection included matching user profile photos to its declared age or address, understanding natural language in user posts, examining the friends of a user, and analyzing the user's IP address and HTTP-related information. For the Facebook dataset, we used the ground-truth of the original study [7], which we also re-validated for the purpose of this work, as we describe next.

Facebook. The dataset contained public profile pages of 9,646 real users who received friend requests from fake accounts. As the dataset was collected in early 2011, we wanted to verify whether these users are still active on Facebook. Accordingly, we revisited their public profiles in June 2013. We found that 7.9% of these accounts were either disabled by Facebook or deactivated by the users themselves. Accordingly, we excluded these accounts, ending up with 8,888 accounts, out of which 32.4% were victims who accepted a single friend request sent by a fake posing as a stranger. As fakes initially targeted users at random, the dataset included a diverse sample of Facebook users. In particular, these users were 51.3% males and 48.7% females, lived in 1,983 cities across 127 countries, practiced 43 languages, and have used Facebook for 5.4 years on average.

The dataset also included *two graph samples* of Facebook, which were collected using a stochastic version of the Breadth-First Search method called "forest fire" [59]. The first graph consisted of 2,926 real accounts with 9,124 friendships (the real region), 65 fakes with 2,080 friendships (the fake region), and 748 *timestamped* attack edges. The second graph consisted of 6,136 real accounts with 38,144 friendships, which represented the real region only.

Tuenti. The dataset contained profiles of 60K real users who received friend requests from fake accounts, out of which 50% were victims. The dataset was collected in Feb 10, 2014 from live production servers, where data resided in memory and no expensive, back-end queries were made. For Tuenti, collecting this dataset was a low-cost and easy process, as it only involved reading cached user profiles of a subset of its *daily active users*, users who logged in to Tuenti on that particular day.

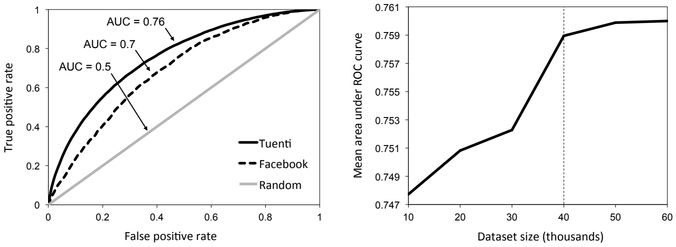
B. Victim prediction

We sought to validate the following claim: An OSN operator can identify unknown victim accounts with a probability that is better than random, using strictly low-cost features extracted from readily-available user profiles.

Features. As described in Table I, we extracted features from both datasets to generate feature vectors. The only requirement we had for feature selection was to have the feature value available for all users in the dataset, so that the resulting feature vectors are complete. For the Facebook dataset, we were able to extract 18 features from public user profiles. For Tuenti, however, the dataset was limited to 14 features, but contained user features that are not publicly accessible.

Validation method. To evaluate the accuracy of the classifiers, we performed a 10-fold, stratified cross-validation method [47] using the RF learning algorithm. First, we randomly partitioned the dataset into 10 equally-sized sets, with each set having the same percentage of victims as the complete dataset. We next trained an RF classifier using 9 sets and tested it using the remaining set. We repeated this procedure 10 times (i.e., folds), with each of the sets being used once for testing. Finally, we combined the results of the folds by computing the mean of their true-positive rate (TPR) and false-positive rate (FPR).

Performance metrics. The output of the classifier depends on its operating threshold, which is a cutoff value in the prediction probability after which the classifier identifies a given user as a victim. In order to capture the trade-off between TPR and FPR in single curve, we repeated the cross-validation method under different threshold values using a procedure known as receiver operating characteristics (ROC) analysis. In ROC analysis, the closer the curve is to the top-left corner at point $(0, 1)$ the better the classification performance is. The quality of the classifier can be quantified with a single value by calculating the *area under its ROC curve* (AUC) [47].



(a) ROC Analysis **(b)** Sensitivity to dataset size

Fig. 4: Victim prediction using the RF algorithm. In (a), the ROC curves show the tradeoff between FPR and TPR for both datasets. In ROC analysis, the closer the curve is to the upper-left corner the more accurate it is. The area under the ROC curve (AUC) summarizes the classifier's performance. Therefore, an AUC of 1 means a perfect classifier, while an AUC of 0.5 means a random classifier. We require the victim classifier to be better than random. In (b), during cross validation on Tuenti dataset, we observed that increasing the dataset size to more than 40K vectors did not significantly increase the AUC.

We also recorded the *relative importance* (RI) of features used for the classification. The RI score is computed by the RF algorithm, and it describes the relative contribution of each feature to the predictability of the label (i.e., a victim or a non-victim), when compared to all other features [46].

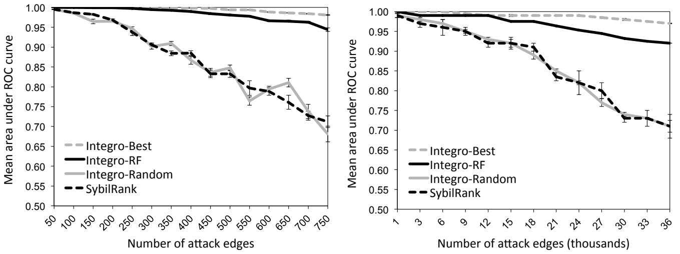
Results. For both datasets, the RF classifier ended up with an AUC greater than 0.5, as shown in Fig. 4a. In particular, for the Facebook dataset, the classifier delivered an AUC of 0.7, which is 40% better than random. For the Tuenti dataset, on the other hand, the classifier delivered an AUC of 0.76, which is 52% better than random. Also, increasing the dataset size to more than 40K feature vectors did not significantly improve the AUC during cross-validation, as show in Fig. 4b. This means an OSN operator can train a victim classifier using a relatively small dataset, so fewer accounts need to be manually verified.

C. Ranking quality

We compared Íntegro against SybilRank in terms of their ranking quality under various attack scenarios, where ideally real accounts should be ranked higher than fake accounts. Our results are based on the average of at least 10 runs, with error bars reporting 95% confidence intervals (CI), when applicable. We picked the Facebook dataset for this comparison because it included both feature vectors and graph samples.

Infiltration scenarios. We considered two attack scenarios. In the first scenario, attackers establish attack edges by targeting users with whom their fakes have mutual friends. Accordingly, we used the first Facebook graph which contained timestamped attack edges, allowing us to replay the infiltration by 65 socialbots (n=2,991 and m=11,952). We refer to this scenario as the *targeted-victim* attack.

In the second scenario, we attackers establish attack edges by targeting users at random [13]. We designated the second



(a) Targeted-victim attack **(b)** Random-victim attack

Fig. 5: The ranking quality of both systems in terms of its AUC under each infiltration scenario (CI=95%). SybilRank and Íntegro resulted in a similar performance when a random victim classifier is used, which represents a practical baseline for Íntegro. As the number of attack edges increased, SybilRank's AUC decreased significantly close to 0.7, while Íntegro sustained its high performance with AUC > 0.9.

Facebook graph as the real region. We then generated a synthetic fake region consisting of 3,068 fakes with 36,816 friendships using the small-world graph model [60]. We then added 35,306 random attack edges between the two regions (n=9,204 and m=110,266). As suggested in related work [34], we used a relatively large number of fakes and attack edges in order to stress-test both systems under evaluation. We refer to the this scenario as the *random-victim* attack.

Propagation rates. For each infiltration scenario, we deployed the previously trained victim classifier in order to assign new edge weights. As we injected fakes in the second scenario, we generated their feature vectors by sampling each feature distribution of fakes from the first scenario.[4] We also assigned edge weights using another victim classifier that simulates two operational modes. In the first mode, the classifier outputs the *best* possible victim predictions with an AUC≈1 and probabilities greater than 0.95. In the second mode, the classifier outputs uniformly *random* predictions with an AUC≈0.5. We used this classifier to evaluate the theoretical best and practical worst case performance of Íntegro.

Evaluation method. To evaluate each system's ranking quality, we ran the system using both infiltration scenarios starting with a single attack edge. We then added another attack edge, according to its timestamp if available, and repeated the experiment. We kept performing this process until there were no more edges to add. At the end of each run, we measured the resulting AUC of each system, as explained next.

Performance metric. For the resulting ranked list of accounts, we performed ROC analysis by moving a pivot point along the list, starting from the bottom. If an account is behind the pivot, we marked it as fake; otherwise, we marked it as real. Given the ground-truth, we measured the TPR and the FPR across the whole list. Finally, we computed the corresponding AUC, which in this case *quantifies the probability that a random real account is ranked higher than a random fake account.*

Seeds and iterations. In order to make the chance of guessing seeds very small, we picked 100 trusted accounts that are non-victim, real accounts. We used a total trust that is equal to n, the number of nodes in the given graph. We also performed $\lceil \log_2(n) \rceil$ iterations for both Íntegro and SybilRank.

Results. Íntegro consistently outperformed SybilRank in ranking quality, especially as the number of attack edges increased. Using the RF classifier, Íntegro resulted in an AUC which is always greater than 0.92, and is up to 30% improvement over SybilRank in each attack scenario, as shown in Fig 5.

In each infiltration scenario, both systems performed well when the number of attack edges was relatively small. In other words, the fakes were sparsely connected to real accounts and so the regions were easily separated. As SybilRank limits the number of fakes that can outrank real accounts by the number of attack edges, its AUC degraded significantly as more attack edges were added to each graph. Íntegro, however, maintained its performance, with at most 0.07 decrease in AUC, even when the number of attack edges was relatively large. Notice that Íntegro performed nearly as good as SybilRank when a random victim classifier was used, but performed much better when the RF classifier was used instead. This shows the impact of leveraging victim prediction on fake account detection.

D. Sensitivity to seed-targeting attacks

Sophisticated attackers might obtain a full or partial knowledge of which accounts are trusted by the OSN operator. As the total trust is initially distributed among these accounts, an attacker can adversely improve the ranking of the fakes by establishing attack edges directly with them. We next evaluate both systems under two variants of this seed-targeting attack.

Attack scenarios. We focus on two main attack scenarios. In the first scenario, the attacker targets accounts that are k *nodes away from all trusted accounts*. This means that the length of the shortest path from any fake account to any trusted account is exactly k+1, representing the *distance* between the seeds and the fake region. For k=0, each trusted account is a victim and located at a distance of 1. We refer to this scenario, which assumes a resourceful attacker, as the *distant-seed* attack.

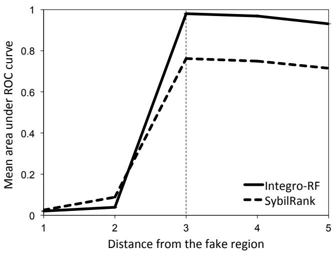
In the second scenario, attackers have only a partial knowledge and target k trusted accounts picked at random. We refer to this scenario as the *random-seed* attack.

Evaluation method. To evaluate the sensitivity of each system to a seed-targeting attack, we used the first Facebook graph to simulate each attack scenario. We implemented this by replacing the endpoint of each attack edge in the real region with a real account picked at random from a set of candidates. For the first scenario, a candidate account is one that is k nodes away from all trusted accounts. For the second scenario, a candidate account is simply any trusted account. We ran experiments for both systems using different values of k and measured the corresponding AUC at the end of each run.

Results. In the first attack scenario, both systems had a poor ranking quality when the distance was small, as illustrated in Fig. 6a. Because Íntegro assigns low weights to edges incident to victim accounts, the trust that escapes to the fake region is less likely to come back into the real region. This explains why SybilRank had a slightly better AUC for distances less than 3. However, once the distance was larger, Íntegro outperformed SybilRank ,as expected from earlier results.

In the second attack scenario, the ranking quality of both systems degraded, as the number of victimized trusted accounts

[4] We excluded the "friends" feature, as it can be computed from the graph.



(a) Distant-seed attack **(b)** Random-seed attack

Fig. 6: The sensitivity of both systems to each seed-targeting attack (CI=95%). In distant-seed attack, an attacker befriends users that are at a particular distance from *all* trusted accounts, which represents a practical worst case scenario for both system. In the random-seed attack, the attacker directly befriends a *subset* of the trusted accounts. Overall, both systems are sensitive to seed-targeting attacks.

increased, where Íntegro consistently outperformed SybilRank, as shown in Fig. 6b. Notice that by selecting a larger number of trusted accounts, it becomes much harder for an attacker to guess which account is trusted, while the gained benefit per victimized trusted account is further reduced.

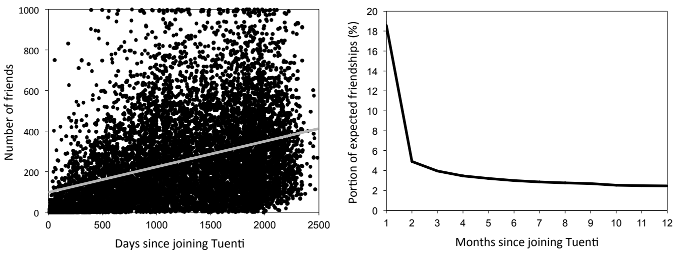
E. Deployment at Tuenti

We deployed both systems on a snapshot of Tuenti's daily active users graph in February 6, 2014. The graph consisted of several million nodes and tens of millions of edges. We had to mask out the exact numbers due to a non-disclosure agreement with Tuenti. After initial analysis of the graph, we found that 96.6% of nodes and 94.2% of edges belonged to one giant connected component (GCC). Therefore, we focused our evaluation on this GCC.

Preprocessing. Using a uniform random sample of 10K users, we found that new users have weak connectivity to others due to the short time they have been on Tuenti, as shown in Fig. 7a. If these users were included in our evaluation, they would end up receiving low ranks, which would lead to false positives.

To overcome this issue, we estimated the period after which users accumulate at least 10% of the average number of friends in Tuenti. To achieve this, we used a uniformly random sample of 10K real users who joined Tuenti over the last 77 months. We divided the users in the sample into buckets representing how long they have been active members. We then calculated the average number of new friendships they made after every other month. As illustrated in Fig. 7b, users accumulated 53% of their friendships during the first 12 months. In addition, 18.6% of friendships were made after one month since joining the network. To this end, we decided to defer the consideration of users who have joined in the last 30 days since Feb 6, 2014, which represented only 1.3% of users in the GCC.

Community detection. We applied the Louvain method on the preprocessed GCC. The method finished quickly after just 5 iterations with a high modularity score of 0.83, where a value of 1 corresponds to a perfect partitioning. In total, we found 42 communities and the largest one consisted of 220,846 nodes. In addition, 15 communities were relatively large containing more than 50K nodes. Tuenti's account analysts verified 0.05% of the nodes in each detected community, and designated them as trusted accounts for both systems.



(a) Users connectivity **(b)** Friendship growth over time

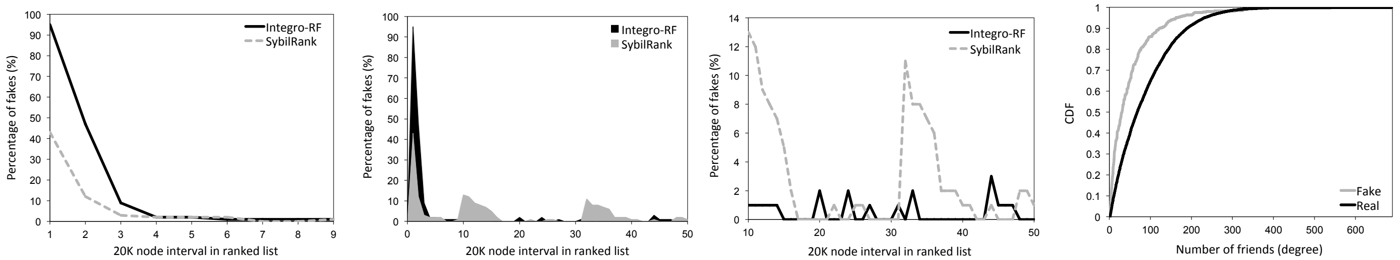
Fig. 7: Preprocessing. In (a), there is a positive correlation between number of days since a user joined Tuenti and how well-connected the user is in terms of number of friends (Pearson's $r = 0.36$). In fact, *93% of all new users who joined Tuenti in the last 30 days had weak connectivity of 46 friends or less*, much smaller than the average of 254 friends. In (b), we found that most of the friendship growth happens in the first month since joining the network, where users on average establish 18.6% of their friendships. We accordingly defer the consideration of users who joined Tuenti in the last 30 days, as they will likely be assigned low ranks.

Performance metric. As the number of users in the processed GCC is large, it was infeasible to manually inspect and label each account. This means that we were unable to evaluate the system using ROC analysis. Instead, we attempted to determine the percentage of fake accounts at equally-sized intervals in the ranked list. We accomplished this in collaboration with Tuenti's analysts by manually inspecting a user sample in each interval in the list. This percentage is directly related to the *precision* of fake account detection, which is a performance metric typically used to measure the ratio of relevant items over the top-k highest ranked items in terms of relevance [61].

Evaluation method. We utilized the previously trained victim classifier in order to weight a copy of the graph. We then ran both systems on two versions of the graph (i.e., weighted and unweighted) for $\lceil \log_2(n) \rceil$ iterations, where n is number of nodes in the graph. After that, we examined the ranked list of each system by inspecting the first lowest-ranked one million users. We randomly selected 100 users out of each 20K user interval for inspection in order to measure the percentage of fakes in the interval, that is, the precision. We do not include the complete range due to confidentiality reasons.

Results. As shown in Fig. 8a, Íntegro resulted in 95% precision in the lowest 20K ranking user accounts, as opposed to 43% by SybilRank and 5% by Tuenti's user-based abuse reporting system. This percentage dropped dramatically as we went up in the list, which means our ranking scheme placed most of the fakes at the bottom of the ranked list, as shown in Fig. 8b.

Let us consider SybilRank's ranking shown in Fig. 8a and Fig. 8c. The precision, starting with 43% for the first interval, gradually decreased until rising again at the 10th interval. This pattern repeated at the 32nd interval as well. We inspected the fake accounts at these intervals and found that they belonged to three different, large communities. In addition, these fakes had a large number of friends, much larger than the average of 254 friends. In particular, the fakes from the 32nd interval



(a) Precision at lower intervals **(b)** Precision over the whole list **(c)** Precision at higher intervals **(d)** Node degree distribution

Fig. 8: Deployment results at Tuenti. The overall ranking quality of both systems is summarized in (b). Ideally, all fake accounts should be in the bottom of the ranked list. In (a) and (c), we observed that Íntegro consistently outperforms SybilRank in term of fake account detection precision (i.e., the percentage of fakes in each sample). In particular, most of the fake accounts identified by Íntegro were located at significantly lower locations in the ranked list, unlike SybilRank. Upon further inspection of fakes at higher intervals, we found that they established a large number of attack edges, as suggested by the degree distribution in (d).

onwards had more than 300 friends, with a maximum of up to 539. Fig. 8d shows the degree distribution for both *verified* fake and real accounts. This figure suggests that fakes tend to create many attack edges with real accounts, which confirms earlier findings on other OSNs such as Facebook [7]. Also, this behavior explains why Íntegro outperformed SybilRank in user ranking quality; these high degree fakes received lower ranks as most of their victims were identified by the classifier.

SybilRank in retrospect. SybilRank was initially evaluated on Tuenti, where it effectively detected a significant percentage of the fakes [13]. The original evaluation, however, pruned excessive edges of nodes that had a degree greater than 800, which include a non-disclosed number of fakes that highly infiltrated Tuenti. Also, the original evaluation was performed on the whole graph, which included many dormant accounts. However, our evaluation was based on the daily active users graph in order to focus on active fake accounts that could be harmful. While this change limited the number of fakes that existed in the graph, it has evidently revealed the ineffectiveness of SybilRank under social infiltration. Additionally, the original evaluation showed that 10–20% of fakes received high ranks, a result we also attest, due to the fact that these fake accounts had established many attack edges. On the other hand, Íntegro has 0–2% of fakes at these high intervals, and so it delivers an order of magnitude better precision than SybilRank.

VI. Implementation and Scalability

We implemented Íntegro in Mahout[5] and Giraph[6], which are widely used, open-source distributed machine learning and graph processing platforms, respectively. We next describe the scalability of Íntegro using a synthetic benchmark.

Benchmark. We deployed Íntegro an Amazon Elastic MapReduce[7] cluster. The cluster consisted of one *m1.small* instance serving as a master node and 32 *m2.4xlarge* instances serving as slave nodes. We employed the small-world graph model [60] to generate 5 graphs with an exponentially increasing number of nodes. For each one of these graphs, we used the Facebook dataset to randomly generate all feature vectors with the same distribution for each feature. We then ran Íntegro on each of the generated graphs and measured its execution time.

Results. Íntegro achieved a nearly linear scalability with the number of nodes in a graph, as illustrated in Fig. 9. Excluding the time required to load the 160M node graph into memory, 20 minutes for a non-optimized data format, it takes less than 2 minutes to train an RF classifier and compute vulnerability scores for nodes, and less than 25 minutes to weight the graph, rank nodes, and finally sort them. This makes Íntegro computationally practical even for large OSNs such as Facebook.

VII. Discussion

As mentioned in Section IV-F, Íntegro's security guarantee is sensitive to the performance of the deployed victim classifier, which is formally captured by the volume $\mathrm{vol}(E_a)$ in the bound $O(\mathrm{vol}(E_a)\log n)$, and can be practically measured by its AUC.

Sensitivity to victim classification. As illustrated in Fig. 5, improving the AUC of the victim classifier from random with AUC ≈ 0.5, to actual with AUC$= 0.7$, and finally to best with AUC ≈ 1 consistently improved the resulting ranking in terms of its AUC. Therefore, a higher AUC in victim prediction leads to a higher AUC in user ranking. This is the case because the ROC curve of a victim classifier monotonically increases, so a higher AUC implies a higher true positive rate (TPR). In turn, a higher TPR means more victims are correctly identified, and so more attack edges are assigned lower weights, which evidently leads to a higher AUC in user ranking.

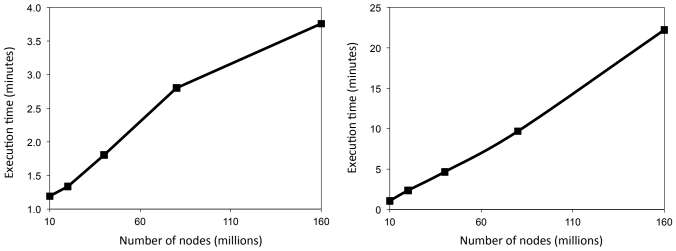
Sensitivity to social infiltration. Regardless of the used victim classifier, the ranking quality decreases as the number of attack edges increases, as illustrated in Fig. 5. This is the case because even a small false negative rate (FNR) in victim classification means more attack edges indecent to misclassified victims are assigned high weights, leading to a lower AUC in user ranking.

Maintenance. While an attacker does not control real accounts nor their activities, it can still trick users into befriending fakes. In order to achieve a high-quality ranking, the victim classifier should be regularly retrained to capture new and changing user behavior in terms of susceptibility to social infiltration. This is, in fact, the case for supervised machine learning when applied to computer security problems [8]. Also, as the ranking scheme

[5] http://mahout.apache.org

[6] http://giraph.apache.org/

[7] http://aws.amazon.com/elasticmapreduce



(a) Mahout **(b)** Giraph

Fig. 9: System scalability on both platforms. In (a), the execution time includes the time to train an RF classifier and compute a vulnerability score for each node in the graph. In (b), the execution time includes the time to weight the graph, rank nodes, and finally sort them.

is sensitive to seed-targeting attacks, the set of trusted accounts should be regularly updated and validated in order to reduce the negative impact of these attacks, even if they are unlikely to occur or succeed in practice, as discussed in Section IV-D.

Impact. By using Íntegro, Tuenti requires nearly 67 man hours to manually validate the 20K lowest ranking user accounts, and discover about 19K fake accounts instead of 8.6K fakes with SybilRank. With its user-based abuse reporting system that has 5% hit rate, and assuming all fakes get reported, Tuenti would need 1,267 man hours instead to discover 19K fake accounts. This improvement has been useful to both Tuenti and its users.

VIII. Limitations

We next outline two design limitations which are inherited from SybilRank [13] and similar ranking schemes [34]:

- *Íntegro's design is limited to only undirected social graphs.* In other words, OSNs whose users declare lateral relationships are not expected to benefit from our proposal. This is the case because directed graphs, in general, have a significantly smaller mixing time than their undirected counterparts [62], which means a random walk on such graphs will converge in a much small number of steps, rendering short random walks unsuitable for robust user ranking.

- *Íntegro delays the consideration of new user accounts.* This means that an OSN operator might miss the chance to detect fakes at their early life-cycle. However, as shown in Figure 7a, only 7% of new users who joined Tuenti in the last month had more than 46 friends. To estimate the number of fakes in new accounts, we picked 100 accounts at random for manual verification. We found that only 6% of these accounts were fake, and the most successful fake account had 103 victims. In practice, the decision of whether to exclude these account is operational, and it depends on the actions taken on low-ranking users. For example, an operator can enforce abuse mitigation technique, as discussed in Section II-C, against low-ranking users, where false positives can negatively affect user experience but slow down fake accounts that just joined the network. This is a security/usability trade-off which we leave to the operator to manage. Alternatively, the operator can use fake account detection systems that are designed to admit legitimate new users using, for example, a vouching process [63].

Íntegro is not a stand-alone fake account detection system. It is intended to complement existing abuse detection systems and is designed to detect automated fake accounts that befriend many victims for subsequent attacks. Íntegro has been deployed at Tuenti along side a feature-based detection system and a user-based abuse reporting system.

IX. Conclusion

OSNs today are faced with the problem of detecting fake accounts in a highly adversarial environment. The problem is becoming more challenging as such accounts have become sophisticated in cloaking their operation with patterns resembling real user behavior. In this paper, we presented Íntegro, a scalable defense system that helps OSN operators to detect fake accounts using a meaningful user ranking scheme.

Our evaluation results show that SybilRank, the state-of-the-art in fake account detection, is ineffective when the fakes infiltrate the target OSN by befriending a large number of real users. Íntegro, however, has proven more resilient to this effect by leveraging in a novel way the knowledge of benign victim accounts that befriend fakes . We have implemented Íntegro on top of standard data processing platforms, Mahout and Giraph, which are scalable and easy to deploy in modern data centers. In fact, Tuenti, the largest OSN in Spain with more than 15M active users, has deployed our system in production to thwart fakes in the wild with at least 10 time more precision.

X. Acknowledgment

We would like to thank our shepherd, Gianluca Stringhini, and our colleagues for their help and feedback on an earlier version of this paper. The first author is thankful to the University of British Columbia for a generous doctoral fellowship.

References

[1] J. R. Douceur, "The sybil attack," in *1st International Workshop on Peer-to-Peer Systems*. Springer-Verlag, 2002, pp. 251–260.

[2] Facebook, "Quarterly earning reports," Jan 2014. [Online]. Available: http://goo.gl/YujtO

[3] CBC, "Facebook shares drop on news of fake accounts," Aug 2012. [Online]. Available: http://goo.gl/6s5FKL

[4] K. Thomas and et al., "Suspended accounts in retrospect: an analysis of Twitter spam," in *Proc. IMC'11*. ACM, 2011, pp. 243–258.

[5] G. Yan and et al., "Malware propagation in online social networks: nature, dynamics, and defense implications," in *Proc. ASIACCS'11*. ACM, 2011, pp. 196–206.

[6] J. Ratkiewicz and et al., "Truthy: mapping the spread of astroturf in microblog streams," in *Proc. WWW'11*. ACM, 2011, pp. 249–252.

[7] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, "The socialbot network: when bots socialize for fame and money," in *Proc. ACSAC'11*. ACM, 2011, pp. 93–102.

[8] T. Stein, E. Chen, and K. Mangla, "Facebook immune system," in *Proceedings of the 4th Workshop on Social Network Systems*. ACM, 2011, pp. 8–14.

[9] L. Alvisi, A. Clement, A. Epasto, U. Sapienza, S. Lattanzi, and A. Panconesi, "SoK: The evolution of sybil defense via social networks," *In Proceedings of the IEEE Symposium on Security and Privacy*, 2013.

[10] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda, "All your contacts are belong to us: automated identity theft attacks on social networks," in *Proc. WWW'09*. ACM, 2009, pp. 551–560.

[11] C. Wagner, S. Mitter, C. Körner, and M. Strohmaier, "When social bots attack: Modeling susceptibility of users in online social networks," in *WWW Workshop on Making Sense of Microposts*, vol. 12, 2012.

[12] M. N. Ko, G. P. Cheek, M. Shehab, and R. Sandhu, "Social-networks connect services," *Computer*, vol. 43, no. 8, pp. 37–43, 2010.

[13] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, “Aiding the detection of fake accounts in large scale social online services,” in *Proc. NSDI’12*. USENIX Association, 2012, pp. 15–15.

[14] S. Yardi, N. Feamster, and A. Bruckman, “Photo-based authentication using social networks,” in *Proceedings of the first workshop on Online social networks*. ACM, 2008, pp. 55–60.

[15] S. D. Kamvar and et al., “The EigenTrust algorithm for reputation management in P2P networks,” in *Proceedings of 12th international conference on World Wide Web*. ACM, 2003, pp. 640–651.

[16] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, “Sybilguard: defending against sybil attacks via social networks,” *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 267–278, 2006.

[17] H. Yu and et al., “Sybillimit: A near-optimal social network defense against sybil attacks,” in *Proc. S&P’08*. IEEE, 2008, pp. 3–17.

[18] G. Danezis and P. Mittal, “Sybilinfer: Detecting sybil nodes using social networks.” in *Proceedings of the 9th Annual Network & Distributed System Security Symposium*. ACM, 2009.

[19] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove, “An analysis of social network-based sybil defenses,” in *Proceedings of ACM SIGCOMM Computer Communication Review*. ACM, 2010, pp. 363–374.

[20] N. Tran, J. Li, L. Subramanian, and S. S. Chow, “Optimal sybil-resilient node admission control,” in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 3218–3226.

[21] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” *Comm. of ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[22] G. Malewicz and et al., “Pregel: a system for large-scale graph processing,” in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 2010, pp. 135–146.

[23] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, “Design and analysis of a social botnet,” *Computer Networks*, vol. 57, no. 2, pp. 556–578, 2013.

[24] T. Hwang, I. Pearce, and M. Nanis, “Socialbots: Voices from the fronts,” *interactions*, vol. 19, no. 2, pp. 38–45, 2012.

[25] M. Egele and et al., “COMPA: Detecting compromised accounts on social networks.” in *Proc. NDSS’13*, 2013.

[26] M. Motoyama and et al., “Dirty jobs: The role of freelance labor in web service abuse,” in *Proceedings of the 20th USENIX Security Symposium*. USENIX Association, 2011, pp. 14–14.

[27] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai, “Uncovering social network sybils in the wild,” in *Proceedings of 2011 ACM Internet Measurement Csonference*. ACM, 2011, pp. 259–268.

[28] G. Stringhini, C. Kruegel, and G. Vigna, “Detecting spammers on social networks,” in *Proceedings of the 26th Annual Computer Security Applications Conference*. ACM, 2010, pp. 1–9.

[29] G. Wang and et al., “You are how you click: Clickstream analysis for sybil detection,” in *Proceedings of the 22nd USENIX Security Symposium*. USENIX Association, 2013, pp. 1–8.

[30] G. Karypis and V. Kumar, “Multilevel k-way partitioning scheme for irregular graphs,” *Journal of Parallel and Distributed computing*, vol. 48, no. 1, pp. 96–129, 1998.

[31] J. Tygar, “Adversarial machine learning.” *IEEE Internet Computing*, vol. 15, no. 5, 2011.

[32] D. Lowd and C. Meek, “Adversarial learning,” in *Proceedings of the 11th ACM SIGKDD*. ACM, 2005, pp. 641–647.

[33] Y. Boshmaf and et al., “Key challenges in defending against malicious socialbots,” in *Proc. LEET’12*, vol. 12, 2012.

[34] H. Yu, “Sybil defenses via social networks: a tutorial and survey,” *ACM SIGACT News*, vol. 42, no. 3, pp. 80–101, 2011.

[35] B. Viswanath and et al., “Exploring the design space of social network-based sybil defenses,” in *In Proceedings of the 4th International Conference on Communication Systems and Networks*. IEEE, 2012, pp. 1–8.

[36] Y. Boshmaf and et al., “Graph-based sybil detection in social and information systems,” in *Proc. ASONAM’13*. IEEE, 2013.

[37] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney, “Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters,” *Internet Mathematics*, vol. 6, no. 1, pp. 29–123, 2009.

[38] S. Fortunato, “Community detection in graphs,” *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.

[39] A. Mislove and et al., “You are who you know: inferring user profiles in online social networks,” in *Proc. WSDM’10*. ACM, 2010, pp. 251–260.

[40] G. Wang, M. Mohanlal, C. Wilson, X. Wang, M. Metzger, H. Zheng, and B. Y. Zhao, “Social turing tests: Crowdsourcing sybil detection,” in *Proc. NDSS’13*. ACM, 2013.

[41] S. Ghosh and et al., “Understanding and combating link farming in the twitter social network,” in *Proceedings of 21st international conference on World Wide Web*. ACM, 2012, pp. 61–70.

[42] A. Elyashar, M. Fire, D. Kagan, and Y. Elovici, “Homing socialbots: intrusion on a specific organization’s employee using socialbots,” in *Proc. ASONAM’13*. ACM, 2013, pp. 1358–1365.

[43] G. Stringhini and et al., “Follow the green: growth and dynamics in twitter follower markets,” in *Proc. IMC’13*. ACM, 2013, pp. 163–176.

[44] C. Yang and et al., “Analyzing spammers’ social networks for fun and profit: a case study of cyber criminal ecosystem on twitter,” in *Proc. of WWW’12*. ACM, 2012, pp. 71–80.

[45] D. A. Spielman and S.-H. Teng, “Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems,” in *Proc. TC’04*. ACM, 2004, pp. 81–90.

[46] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[47] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: Data mining, inference, and prediction, second edition*. Springer, 2009.

[48] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, “Combating web spam with trustrank,” in *Proceedings of VLDB*, 2004, pp. 576–587.

[49] G. H. Golub and H. A. Van der Vorst, “Eigenvalue computation in the 20th century,” *Journal of Computational and Applied Mathematics*, vol. 123, no. 1, pp. 35–65, 2000.

[50] E. Behrends, *Introduction to Markov chains with special emphasis on rapid mixing*. Vieweg, 2000, vol. 228.

[51] M. Dellamico and Y. Roudier, “A measurement of mixing time in social networks,” in *Proceedings of the 5th International Workshop on Security and Trust Management, Saint Malo, France*, 2009.

[52] A. Mohaisen, A. Yun, and Y. Kim, “Measuring the mixing time of social graphs,” in *Proceedings of the 10th annual conference on Internet measurement*. ACM, 2010, pp. 383–389.

[53] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, “Statistical properties of community structure in large social and information networks,” in *Proc. WWW’08*. ACM, 2008, pp. 695–704.

[54] V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, 2008.

[55] M. E. Newman, “Modularity and community structure in networks,” *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[56] Y. Boshmaf, D. Logothetis, G. Siganos, J. Lería, J. Lorenzo, M. Ripeanu, and K. Beznosov, “́Integro: Leveraging victim prediction for robust fake account detection in OSNs,” *LERSSE technical report*, 2014.

[57] A. Sinclair, “Improved bounds for mixing rates of Markov chains and multicommodity flow,” in *Proceedings of Latin American Symposium on Theoretical Informatics*. Springer-Verlag, 1992, pp. 474–487.

[58] D. N. Tran, B. Min, J. Li, and L. Subramanian, “Sybil-resilient online content voting.” in *NSDI*, vol. 9, 2009, pp. 15–28.

[59] J. Leskovec and C. Faloutsos, “Sampling from large graphs,” in *Proceedings of the ACM SIGKDD Conference*. ACM, 2006, pp. 631–636.

[60] D. J. Watts and S. H. Strogatz, “Collective dynamics of small-world networks,” *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[61] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, “Evaluating collaborative filtering recommender systems,” *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 5–53, 2004.

[62] A. Mohaisen, H. Tran, N. Hopper, and Y. Kim, “On the mixing time of directed social graphs and security implications,” in *Proceedings of the ASIACCS Conference*. ACM, 2012, pp. 36–37.

[63] Y. Xie and et al., “Innocent by association: early recognition of legitimate users,” in *Proc. CCS’12*. ACM, 2012, pp. 353–364.