Defending Against Large-scale Crawls in Online Social Networks

Mainack Mondal, Bimal Viswanath, Allen Clement,
Peter Druschel, Krishna P. Gummadi, Alan Mislove[†], Ansley Post

MPI-SWS
{mainack, bviswana, aclement, druschel, gummadi, abpost}@mpi-sws.org

[†]Northeastern University
amislove@ccs.neu.edu

ABSTRACT

Thwarting large-scale crawls of user profiles in online social networks (OSNs) like Facebook and Renren is in the interest of both the users and the operators of these sites. OSN users wish to maintain control over their personal information, and OSN operators wish to protect their business assets and reputation. Existing rate-limiting techniques are ineffective against crawlers with many accounts, be they fake accounts (also known as Sybils) or compromised accounts of real users obtained on the black market.

We propose Genie, a system that can be deployed by OSN operators to defend against crawlers in large-scale OSNs. Genie exploits the fact that the browsing patterns of honest users and crawlers are very different: even a crawler with access to many accounts needs to make many more profile views per account than an honest user, and view profiles of users that are more distant in the social network. Experiments using real-world data gathered from a popular OSN show that Genie frustrates large-scale crawling while rarely impacting honest users; the few honest users who are affected can recover easily by adding a few friend links.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*

General Terms

Security, Design, Algorithms

Keywords

Sybil attacks, social networks, network-based Sybil defense

1. INTRODUCTION

Online social networking sites (OSNs) like Facebook, MySpace, and Orkut have the personal data of hundreds of millions of users. OSNs allow users to browse the (public) profiles of other users in the network, making it easy for users to connect, communicate, and share content. Unfortunately, this functionality can be exploited by third-parties to aggregate and extract data about millions of OSN users. Once collected, the data can be re-published [20], monetized, and mined in ways that may violate users' privacy. For instance, it has been shown that private user attributes like sexual orientation can be inferred from a user's set of friends and their profile attributes [20, 29]; a third party with access to aggregated user data could easily apply these techniques.

These third-party aggregators, which we refer to as *crawlers*, represent a significant problem for OSN site operators as well. User data provides OSN operators with a revenue stream (e.g., via targeted advertisements); stopping crawlers is therefore in the OSN operators' business interests. Additionally, OSN operators cannot ensure that data collected by a third party is used according to the operator's privacy policy. Yet, OSN operators are likely to be held responsible if crawled data is used in ways that violate the policy, at least in the court of public opinion. For example, Facebook was widely blamed in the popular press [12] when a single crawler gathered public profiles of over 100 million users [20]. Thus, OSN operators need effective mechanisms to thwart large-scale crawling of OSN sites [40][1].

Today, OSN operators typically limit the rate at which a single user account or IP address can view user profiles [41], in order to discourage large-scale data collection. Unfortunately, crawlers can circumvent these schemes by creating a large number of fake user accounts [3], by employing botnets [21] or cloud services [8] to gain access to many IP addresses, or by using the compromised accounts of a large number of real users [1].

In this paper, we propose Genie, a system that OSN operators can deploy to limit large-scale crawlers. Genie leverages the differences in the browsing patterns of honest users and crawlers to effectively thwart large-scale crawls of user profiles. Genie's design is based on the insight that honest users tend to view the profiles of others who are well connected and close in the social network. A crawler, on the other hand, is limited in his ability to form or control enough links to be close to all users whose profiles he wishes to view. Genie exploits this fact by enforcing rate limits in

CoNEXT'12, December 10–13, 2012, Nice, France.

[1]Not all large-scale crawls are for nefarious purposes. Researchers, for instance, already tend to obtain the consent of the OSN operator before doing their crawls for research purposes (e.g., [9, 22]); such authorized crawlers can be whitelisted even if a defense is in place.

a way that is sensitive to the distance and degree of connectivity between viewer and viewee.

Using profile view data from RenRen, a Facebook-like OSN that is popular in China [39], we observe that average social network distance between honest users and the profiles they view tends to be low (1.62 social network hops). We demonstrate that a crawler, on the other hand, would require a very large number of well-distributed accounts (e.g., controlling 3% of all existing accounts in OSN) to be able to view the profiles of all users while maintaining the same low average distance.

Genie works by deriving a credit network [10, 14, 15] from the social network. In brief, credit is associated with links in the social network, and a viewer must "pay" credits to the viewee, along a path in the social network, when viewing a profile. Compared to conventional per-account or per-IP address rate-limiting, credit networks offers two key advantages. First, in a credit network, the rate limits are associated with social network links rather than user accounts or addresses. As a result, a crawler gains little by creating many Sybil accounts or using many IP addresses [22]. Second, the greater the distance between viewer and viewee in the social network, the stricter the rate limit is imposed by the credit network on profile views. Consequently, even crawlers with access to a relatively large number of compromised user accounts are unable to crawl the network quickly.

The contributions of this work are as follows:

- We analyze profile viewing data from the Renren social network and show that the average distance in the social network between a honest viewer and a viewee is significantly smaller than that of a crawler. Moreover, a crawler interested in crawling the entire social network is fundamentally unable to blend in with honest viewers unless he controls a very large proportion of strategically positioned user accounts.
- We present the design of Genie, which leverages credit networks derived from the already-existing social network to block large-scale crawling activity, while allowing honest users' browsing unhindered.
- We demonstrate the feasibility of deploying Genie with an evaluation using large partial social network graphs obtained from Renren, Facebook, YouTube, and Flickr, and a mix of real and synthetically generated profile viewing traces. We demonstrate that Genie effectively blocks crawlers while the impact on honest users is minimal.
- We show that Genie can scale to networks having millions of nodes by scaling up credit network operations. Thus, Genie is practical on the large OSNs of today.

2. RELATED WORK

In this section, we describe relevant prior work on limiting large-scale crawls of OSNs and leveraging social networks to defend against Sybil attacks.

Limiting large-scale crawlers There exist a number of techniques that aim to prevent crawls of web services. Two techniques commonly used in practice are `robots.txt` [4], and IP-address- or account-based rate limiting [18, 38, 46].

`robots.txt` is a file stored at the web server that indicates a set of pages that should not be crawled. Compliance with this policy is voluntary; `robots.txt` consequently provides little defense against malicious crawlers.

Large websites like Yahoo! often rely on a simple per-IP-address rate limit to control access to their web services [38]. Each IP address is allocated a maximum number of requests, which are replenished in 24 hour intervals. Once a user exceeds this limit, the operator either stops serving the user or may require that the user solve a CAPTCHA [5]. This approach limits the number of views a crawler can perform from an individual IP address, but is not effective against botnets that control many IPs. Additionally, dedicated crawlers can bypass defenses like CAPTCHA using available CAPTCHA-solving service providers [32], and other schemes exist that can bypass IP-address-based rate-limiting approaches [8, 16].

Online social networks like Facebook, Google Plus or Twitter [18, 41, 46] often use account-based rate limits on requests to view profile pages. Similar to IP-based rate limits, this approach works well if crawlers control at most a small number of accounts; in the face of Sybils or compromised accounts, it is not effective.

Wilson et al. proposed SpikeStrip [52], a system designed to discourage OSN crawlers. SpikeStrip uses cryptography to make information aggregation from OSN websites inefficient. SpikeStrip rate limits the number of profile views allowed per browsing session and prevents different browsing sessions from sharing data. Thus, crawlers cannot aggregate or correlate data gathered by different sessions.

Despite its elegant design, SpikeStrip restricts the functionality of the OSN. For example, SpikeStrip does not allow two OSN users to share website links of a common friend. Moreover, SpikeStrip would require OSNs like Facebook to change the way they use content distribution networks like Akamai to serve users' content. Unlike SpikeStrip, Genie does not affect the OSN functionality or content distribution. As we will show later, Genie can be deployed with minimal disruption to the browsing activities of honest users.

Social network-based Sybil defenses Recently there have been proposals to leverage social networks to defend against Sybil attacks [11, 28, 35, 37, 44, 45, 53, 54].

Sybil defense proposals such as SybilLimit [53] or SybilInfer [11] try to detect Sybils in the network and then block them from the service. However these Sybil detection mechanisms are not designed to address compromised accounts. Additionally, recent work has shown that these Sybil detection schemes suffer from limitations due to assumptions they make concerning the structure of the social network [30, 50].

The design of Genie borrows credit network [10, 14, 15] techniques from Sybil-tolerant [48] systems like Ostra [28] and Bazaar [35], and uses Canal [49] to manage credit network operations. In contrast to Sybil detection schemes, Sybil tolerant systems do not aim to detect Sybil users; instead they minimize the impact of Sybils on honest users.

Genie differs from existing Sybil tolerant systems in two fundamental ways: First, Genie considers crawlers that have access to both Sybil and compromised accounts; previous work considered only Sybil attacks. Second, unlike Ostra and Bazaar (which rely on users to provide feedback on whether a communication is spam or whether a transaction is fraudulent), Genie infers whether or not activity is malicious by exploiting differences in the browsing patterns of crawlers and honest users.

3. SYSTEM AND ATTACK MODEL

3.1 System model

OSN sites such as Facebook, Renren [39], Google+, and Orkut share a common system model: Users create accounts, establish friend links with other users, and post content (often of a personal nature). Users have a "home page" on the OSN that links to all of the user's content; we refer to this as the user's *profile*. The graph formed by the entire set of friend links forms a social network. In the OSNs of interest to us, forming a friend link requires the consent of both users.

Users can typically choose to make their data *private* (i.e., visible only to the user and the site operator), *public* (i.e., visible to every user of the social network), or *semi-public* (i.e., visible to subsets of the user's friends or to friends of user's friends). In practice, many users choose to make their profile information public [24], despite the private nature of some of the information posted. Contributing to this choice may be that sites encourage public sharing [43], that the default privacy setting is "public" [36], that other privacy choices are not always intuitive [42], and that many users are not fully aware of the privacy risks [24]. It is these public profiles (that typically represent a large fraction of all user profiles [23]) that Genie is concerned with protecting.

3.2 Attack model

Today, social networking sites tend to impose a rate limit on the profile views a single user can request, in order to slow down crawlers. However, there are two ways in which a crawler can overcome these limits.

A crawler can conduct a *Sybil attack* by creating multiple user accounts, thereby overcoming the per-user rate limit. It is important to note that while the crawler can create an arbitrary number of links between Sybil accounts he controls, we assume that his ability to form links between his Sybil accounts and honest users is limited by his ability to convince honest users to accept a friend link, regardless of how many user accounts he controls. The significance of this point will become clear in the following section, where we describe how Genie leverages social links to limit crawling activity.

A crawler can also conduct a *compromised account attack* by taking control (e.g., by obtaining the password) of existing accounts in the system. The crawler can gain access to such accounts via phishing attacks, by guessing the user's password, or by purchasing the credentials of already compromised accounts on the black market. A compromised account attack is more powerful than a Sybil attack, because every additional compromised account increases the number of links to honest users that the crawler has access to. Again, the significance of having access to such links will become clear in the following section.

We assume that a crawler with access to compromised accounts cannot compromise the accounts of strategically positioned users of his choosing. Defending against a crawler who can access any account of his choosing would require preventing social engineering attacks, which are outside Genie's attack model. Instead, we assume that compromised accounts are randomly distributed throughout the network. Additionally, we assume that the crawler does not actively form new links involving compromised accounts, as such activity would likely alert the actual owner of the account that their account has been compromised.

We are concerned about attacks where the crawler greedily attempts to gather as many distinct user profiles as possible. We assume that the crawler is agnostic to *which* users he crawls. Our crawler model captures both third-party crawlers [2] as well as research-oriented crawlers (e.g., used in studies of Facebook [22], Flickr [27], and Twitter [9]). However, our model excludes some crawlers that may be interested in repeatedly crawling the accounts of a small subset of users over an extended period of time, perhaps to gather their changing profile information. We make no assumptions about the crawler's strategy, i.e., whether the crawler employs random walks or BFS or DFS to fetch user profiles. Consequently, we simulate attacks employing the strategy that optimizes for the crawler's goal of fetching as many distinct user profiles as possible.

4. WORKLOAD ANALYSIS

In this section, we compare profile viewing workloads of honest users and crawlers. In later sections, we show how Genie can exploit the differences in the browsing behavior of honest users and crawlers to rate-limit crawlers, while rarely affecting honest users.

4.1 Honest users' profile viewing workload

We obtained anonymized user profile browsing data [22] from the RenRen social network [39], a Facebook-like social network that is popular in China. The data covers users in RenRen's Peking University (RR-PKU) network, and includes the links between PKU users and all other RenRen users. We pre-processed the social network and browsing trace to only include the subgraph of the PKU users, and then extracted the largest connected component (LCC) from the social network. Similar to prior work [28, 49], our analysis only examines users in the LCC (representing 91.2% of the users and 94.3% of the links from the pre-processed network). The LCC of the RR-PKU network has 33,294 users and 705,248 undirected links.

The data set also includes a trace of all profiles (both friends and non-friends) that each user browsed during a two-week period during September, 2009. Unfortunately, the RenRen trace does not provide timestamps or an ordering for profile views. Therefore, in experiments where we need the profile views to be ordered, we generate a time series by assigning each profile view a timestamp chosen uniform randomly within the two-week period covered by the trace. This time series was used in all analyses conducted in the paper. We highlight our key findings below.

1. Most users make (receive) few profile views, but a small number of users make (receive) a large number of views. Figure 1 shows the distribution of profile views made and received by individual users in the RR-PKU network. The plots show a considerable skew in the distributions: Most ($> 90\%$) users make or receive fewer than 10 views, while a handful of users ($< 0.4\%$) view 50 or more profiles. In particular, there are three users who viewed 1,827, 612 and 272 profiles (respectively) over a period of two weeks. These users show significant crawler-like behavior, and we return to discuss these users in Section 6.4. Thus, most users in the social network tend to make

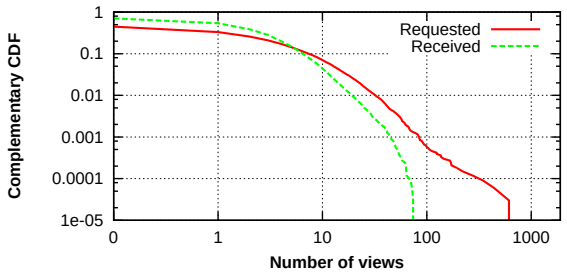

Figure 1: Complementary cumulative distribution of the number of profile view requests made and received by RR-PKU users during a two-week period.

Figure 2: Cumulative distribution of hop distance separating viewers and viewees in RR-PKU network. The profile viewing activities are highly local.

or receive views from a small number of other users in the network.

2. The number of profile views made (received) by users is significantly correlated with their degree. The Pearson correlation coefficients between the rankings of users ordered based on the number of views they make (receive) and their degree is 0.67 (0.5). The high correlation coefficient affirms the intuitive hypothesis that users who are more active in the social network would also have more friends in the network. This finding is consistent with the profile browsing behavior previously studied in Facebook [6] and Orkut [7].

This observation suggests that the *imbalance* in user activity (defined to be the difference between the number of profile views made and received by a user, divided by their degree) would be tightly bounded. We find this to be true: over 99.9% of users have an imbalance lower than 10, with a median value of 0.05.

3. Not all profiles views are unique; a small but non-trivial number of views are repeated. Users tend to repeatedly visit the profiles of others to track updates. In our two-week trace, we found 17,307 (17.8%) of the profile views to be such repeat views. Our estimate of the fraction that repeat views represent is likely to be conservative, as we are restricted to a two-week trace. (One would expect the percent of repeat views to increase with the length of the workload trace.) However, the implication of the presence of repeat views is that repeat views decrease the number of distinct profiles viewed by users. For crawlers, repeat views represent sub-optimal use of resources, as their goal is to view the profiles of as many distinct users as possible.

4. Users tend to make (receive) profile views of others who are within their immediate (1 or 2-hop) network neighborhood. Figure 2 shows the distribution of network distance, measured in terms of hops, between the viewers and viewees in our RR-PKU trace. We observe that over 80% of all profile views are between users who are separated by no more than two hops. This observation is consistent with prior studies of the Orkut social network [7], as well as studies of friendship formation in the Flickr social network [26].

Network	Users	Links	Average degree
RR-PKU [22]	33,294	705,262	21.2
Facebook [47]	63,392	816,886	25.7
Youtube [27]	1,134,889	2,987,624	5.2
Flickr [26]	1,624,991	15,476,835	19.0

Table 1: Number of users, links, and average user degree in the large-scale social networks used to evaluate Genie.

Our analysis considers only a single network, due to the difficulty in obtaining detailed profile viewing data. However, we note that many of our findings are consistent with prior studies of other social networks [6,7,26,51], suggesting that our RenRen data is likely to be representative of other social networks.

4.2 Crawlers' profile viewing workload

We now turn to examining the workload that a crawler would generate when run on the RenRen network from the previous section and three others: the Facebook New Orleans regional network [47], YouTube [27], and Flickr [26]. Table 1 provides more details on the number of users, links, and average degree in these networks.

We simulate crawlers of varying strength by allowing crawlers to "compromise" 1, 10, 100, and 1,000 randomly chosen users within the network. Table 2 shows the number of links that connect user accounts under the crawlers' control to honest users in the different graphs. Note that while a crawler with 1,000 compromised accounts might not seem particularly strong, it is important to consider the size of the networks. For example, the crawler controlling 1,000 accounts controls around 0.1% of all users in the YouTube network. As a point of reference, this would be equivalent to controlling 1,000,000 compromised accounts in the real-world Facebook network.

To generate the crawling workload, we implement the fol-

	Number of compromised accounts			
Network	1	10	100	1000
RR-PKU [22]	26	415	3,638	14,938
Facebook [47]	13	237	2,123	15,970
Youtube [27]	6	26	592	4,129
Flickr [26]	5	613	2,242	16,015

Table 2: Strength of crawlers in different social graphs. Each column corresponds to specific number of random compromised accounts, and the corresponding number of attack links in different graphs.

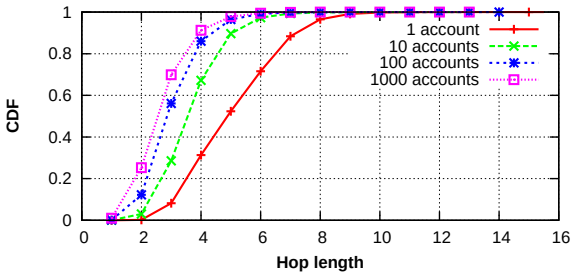

Figure 3: Cumulative distribution of the distance of crawler profile views in Flickr with different numbers of compromised accounts. To fully mimic the high locality in honest user views the crawlers have to control more than 1,000 compromised honest user accounts.

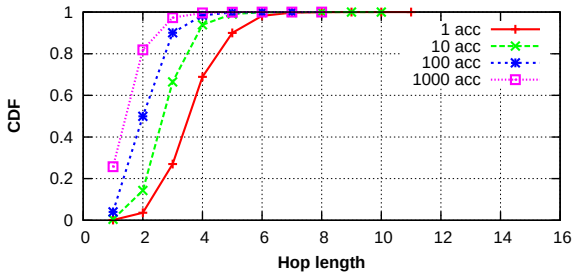

Figure 4: Cumulative distribution of the distance of crawler profile views in Facebook with different numbers of compromised accounts.

lowing strategy for the crawler: We assume that all crawler's accounts collude to view profiles of all honest users in the network. Each honest user is viewed only once, and the crawler's account nearest to an honest user will be assigned the task of viewing that user's profile. This strategy maximizes locality in profile views, making the crawler's workload look as "close" to the honest users' workload as possible.

We now compare the resulting crawler workload with that of honest users, noting two important differences.

1. In contrast to honest users' workload, profile views by crawlers are highly non-local. Figure 3 and Figure 4 shows the locality in profile visits by crawlers with different attacking capacities for the Flickr and Facebook graphs (the other graphs show similar behavior and are removed for brevity). For large network graphs like the Flickr and YouTube samples, we observe that even a powerful crawler with 1,000 accounts has only a small fraction (less than 30%) of requested profiles within the 2-hop neighborhood of the users under his control. For small network graphs like the Facebook and RenRen samples, the crawler does have a majority of users (around 80-90%) within a 2-hop neighborhood. However, in these networks, 1,000 compromised accounts represent 3% of all users; considering that controlling 3% would require controlling 29 million user accounts in the current complete Facebook network, this is a very powerful attack indeed.

Overall, the results indicate that to mimic the high locality in profile views for honest users, crawlers would fundamentally have to control a very large fraction of all accounts.

2. In contrast to honest users, crawlers request many more profile views than they receive. We observe that the median imbalance per link in profile views is 8.3 for a crawler with 100 user accounts, compared to honest users' median imbalance of 0.05. Such an imbalance is necessary, as even with 100 accounts, the crawler makes significantly more profile views that an honest user. In the next section, we present a system design that exploits these differences in browsing patterns of honest users and crawlers.

5. GENIE DESIGN

In this section, we present the design of Genie, analyze its security properties, and discuss how Genie can be used to maliciously deny service to honest users.

5.1 Strawman

To motivate the need for Genie's more elaborate approach, we briefly consider a simple distance-based rate-limiting technique as a strawman design, and show that it is ineffective against Sybil crawlers. We know from Section 4.1 that honest users rarely view profiles outside their neighborhood social graph, whereas the crawlers have to view distant profiles. Our strawman uses distance based rate limiting to leverage this finding.

In the strawman design each user account is allowed to view user profiles at a maximal rate r, where viewing a profile K hops away counts as viewing $K-1$ profiles. (Thus, viewing a friend's profile is not subject to rate-limiting.) The scheme discriminates heavily against crawlers, who tend to view distant profiles. However, just like the existing rate-limiting schemes discussed in Section 2, this design is vulnerable to Sybil attacks. A crawler can simply create more Sybil accounts to overcome the per-account rate limit. The same would be true for any per-account or per-IP-address rate-limiting approach, no matter how much it discriminates against workloads typical of crawlers.

To summarize, if the profile viewing privileges are assigned based only on the viewing user, then a crawler can view more profiles simply by creating additional Sybils. Instead, Genie attaches the profile viewing privileges to *paths* in the network, rather than users, as we describe in the following section.

5.2 Genie design overview

Now, we present the design of Genie. Genie relies on a credit network [14,15] to make sure Genie's rate limits cannot be circumvented by using more Sybil accounts. Moreover, Genie uses the credit network to impose rate limits that discriminate against a crawlers' workload, in order to slow down powerful crawlers that use many compromised user accounts.

A credit network is a directed graph $G = (V, E)$, where each edge $(u, v) \in E$ is labeled with a scalar credit value $c_{uv} \geq 0$. Each node in Genie's credit network corresponds to an OSN user and there is a pair of directed edges $(u, v), (v, u)$ in the credit network iff the users u, v are friends in the OSN. Genie allows user s to view user t's profile information iff the max-flow between s and t in the credit network is at least $f(d_{st})$, where f is a non-decreasing cost function, and d_{st} is the length of the shortest path between s and t in the social network.

Thus Genie computes the *amount of credit charged* for

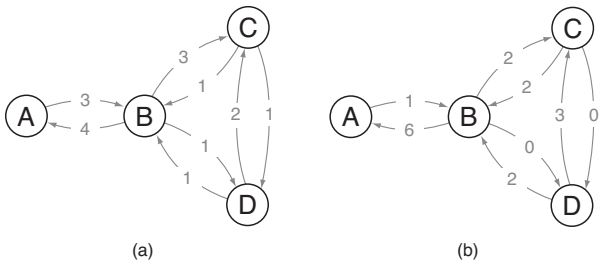

Figure 5: Example of Genie on a small credit network. (a) User A wishes to view D's profile; let us assume this costs two credits. No single path has two credits available, so both $A \to B \to C \to D$ and $A \to B \to D$ are debited 1 credit. (b) The state of the credit network afterwards; note that intermediate nodes B and C maintain the same total credit available, as debiting from one link automatically adds credit to the opposite link.

a profile view based on the shortest path length between the viewer and viewee. However, the credits can actually be exchanged over any set of paths between the viewer and viewee. For example, Figure 5 (a) shows an example of Genie in a small network, where user A wishes to view D's profile. A is charged based on the shortest-path distance (two hops) to D, but the credits may be exchanged over a set of longer, different paths.

If the view is allowed, then the credit value on each link (u, v) on each path p_i that comprises the flow is reduced by c_{p_i}; the credit on each link (v, u) is correspondingly increased by c_{p_i}, where $c_{p_i} \leq f(d_{st})$. An example of this credit adjustment is presented in Figure 5 (b). It is worth noting that Genie rejecting a profile view request does not necessarily mean the operator must block the user making the request; Genie merely flags individual views as suspicious. How the provider responds is a matter of policy—normally, the OSN operator provider would deny or delay the view, effectively slowing down the suspected crawler's activity, but not block the user permanently.

The net effect of this transaction should be that the viewer has $f(d_{st})$ fewer credits available while the viewee has $f(d_{st})$ more credits available for future activity. If j is an intermediate user on one of the paths p_i then the total number of credits available to intermediate user j is unchanged, though the distribution of credit among links adjacent to j is different. As long as user j is well connected and can reach other users through any adjacent link, the change in credit distribution is unlikely to impact the user [10]; if j is not well connected, then attempted views may be flagged due to a lack of *liquidity*.[2]

Social networks exhibit a very high degree of connectivity, ensuring good liquidity in the credit network for most users. New users, inactive users, or small fringe communities that have not (yet) established strong connections to the rest of the network may have issues with liquidity. We will consider this issue in more detail in Section 6.

Genie leverages three key characteristics of honest user activity identified in Section 4 to thwart large-scale crawlers.

[2]Liquidity in the context of credit networks is defined as the capacity to route credit payments [10].

Leveraging unbalanced view ratios We observed in Section 4 that honest users have a balanced ratio of views requested to views received, while crawlers issue many more views than they receive. Genie allows a user to use credits obtained by being viewed to perform views in the future. This ensures liquidity amongst honest users with balanced activity ratios while draining credits from crawlers.

Because honest users do not have a perfect balance of views, even honest users run the risk of eventually exhausting all of their credit. To address this concern, Genie rebalances the credits on a pair of links $(u, v), (v, u)$ at a fixed rate r_b. For example, this can be implemented by dividing time into intervals, and at the beginning of each interval

$$c_{uv} \leftarrow c_{uv} - \tfrac{r_b}{2}(c_{uv} - c_{vu})$$
$$c_{vu} \leftarrow c_{vu} + \tfrac{r_b}{2}(c_{uv} - c_{vu})$$

where $0 < r_b \leq 1$.

Leveraging different path lengths Honest users tend to view profiles of other users that are nearby in the social network; crawling a significant fraction of the social network requires crawlers to view users who are disproportionately far from the crawler in the social network. Genie discriminates against crawlers by charging more credits to view distant users. The cost, in credits, to view a user that is distance d_{st} away in the social network is defined by the simple cost function $f(d_{st}) = d_{st} - 1$.

Leveraging repeated views Honest users repeatedly view the same subset of profiles in the network; crawlers eschew repeat views unless they are re-crawling the network to track changes. Genie does not charge for repeat views that occur within a given time period. If user s views the profile of user t within T days of when s was last charged for viewing t then s is not charged for the profile view; if more than T days have elapsed then s is charged as normal. A typical value of T would be on the order of months.

5.3 Security properties

We now describe the security properties provided by Genie.

Let $C \in V$ be the set of user accounts controlled by the crawler and $H = V \setminus C$ be the set of user accounts not controlled by the crawler (i.e., the honest users). The crawler's goal is to view the profiles of all users in H as quickly as possible. (He can trivially obtain the profiles of users in C.)

We call a link (c, h) in the social network an *attack link* if $c \in C$ and $h \in H$. The cut separating C and H (i.e., the set of attack links) is called the *attack cut*.

To determine the rate r_c at which a crawler can view profiles in H, we need only consider the attack cut, because all of the crawler's views have to cross this cut. Profile views within H or within C are irrelevant, because they must cross the attack cut an even number of times, and do not change the credit available along the cut.

The rate r_c is determined by the following factors:

- A, the size of the attack cut (number of attack links): a powerful crawler has a large attack cut.
- d_h, the average OSN distance for honest profile views.
- d_c, the average OSN distance between users in H and the corresponding closest user in C.
- r_c, the expected rate of profile views received by users in C from users in H: per our threat model, the crawler has little control over this rate, and we can conserva-

tively assume that it is the same as the expected rate of views received by a user in H.
- r_b, the rebalancing rate.
- f, the view cost function.

Using $f(d) = d - 1$, the maximal steady-state crawling rate

$$r_c = A\frac{r_b + r_c(d_h - 1)}{d_c - 1}$$

The numerator is the crawler's "income", the rate at which he can acquire credits. The denominator is the crawler's "cost", in credits, per profile view. As we can see, the maximal crawling rate increases linearly with the number of attack links, at a slope defined by the second term. A larger r_b, r_c or d_h increases, a larger d_c decreases the slope.

The credit network effectively makes the power of a corrupted account attack proportional to the number of acquired attack links.

A crawler can increase the crawling rate by obtaining additional attack links, which are difficult to obtain in large quantities. Obtaining an attack link requires forming a social link with a user not already controlled by the crawler, or compromising a user account that has social links with users not already controlled by the crawler. Creating more user accounts by itself is ineffective, because it does not yield new attack links. As a result, the credit network renders Sybil attacks as such ineffective. Additionally, purchasing many compromised accounts is unlikely to provide the attacker with much additionally crawling ability: many accounts available in underground marketplaces have been observed to not be well connected users, but rather poorly connected users on the fringes of the OSN [33].

5.4 Potential for denial-of-service attacks

Credit exhaustion is a key concern for any credit network-based system where a crawler can consume credits to prevent honest user activity. In the context of Genie we consider two distinct resource exhaustion attacks: First, can a crawler prevent honest users' profile views from taking place? Second, can a crawler target weakly-connected honest users from viewing other profiles or from being viewed? Due to space constraints, we can only summarize the results of our analysis; details can be found in a technical report [31].

A crawler would have to be quite powerful to be able to have a noticeable effect on cuts through the core of the network, which would be necessary for the crawler to impact large numbers of users. However, a modestly strong crawler can impair users in small fringe communities. However, such users can respond by forming additional links to the core of the network.[3] We will further explore the impact of crawlers on honest users empirically in Section 6.

6. EVALUATION

In this section, we evaluate the performance of Genie over several different social networks. When evaluating Genie's performance, we focus on the two primary metrics of interest: (1) the time required for a crawler to crawl a Genie-protected network and (2) the amount of honest users' activity flagged by Genie.

[3]Recall our assumption that a crawler cannot compromise the accounts of specific users of his choosing. If a crawler were able to do this, he could target weakly connected users or small communities very effectively.

Network	Users	Profile views
RR-PKU [22]	33,294	77,501
Facebook [47]	63,392	98,960
Youtube [27]	1,134,889	984,425
Flickr [26]	1,624,991	1,703,831

Table 3: Statistics of synthetic profile view workloads generated for different networks.

6.1 Datasets used

To evaluate the performance of Genie, we need four datasets: (i) a social network graph, (ii) a time-stamped trace of honest users' profile views in the form of (X, Y, t) where user X views user Y's profile at time t, (iii) a crawler's topology (i.e., how the user accounts controlled by the crawler are embedded in the network), and (iv) the crawler's profile crawling trace.

Social network graphs We evaluate the performance of Genie on social network graphs taken from four different online social networks: RenRen [22] (RR-PKU), Facebook [47], YouTube [27] and Flickr [26]), which were introduced in Section 4.2. Table 1 shows their high-level characteristics.

Gathering and generating workload traces Gathering profile viewing traces for large social networks is rather difficult, as it requires explicit cooperation from social network site operators. Unfortunately, many OSN operators are reluctant to share such traces due to competitive and privacy concerns [34]. Thus, we were able to obtain a profile viewing trace for the RR-PKU [22] network only.

As a result, we design a workload generator that reproduces the key features of the original RR-PKU profile viewing trace that we observed in Section 4. We focus on two features that capture the correlation between profile view request/receiver user degree and the number of views per user, and the locality of profile views.

It is difficult to preserve the correlation between both requester and receiver user degrees and number of interactions while ensuring the locality of interactions because of varying degree distribution and path length distribution across different networks. Instead, we generated two synthetic traces: a *receiver trace* that preserves the correlation between the receiver user degree and number of received views while ensuring the locality of interaction, and a *requester trace* that preserves the correlation corresponding to requester user degree and number of requests made while ensuring locality of interaction. Due to space constraints, we only present results for the *receiver trace*. Results for requester traces are similar and are shown in our extended technical report [31]. The high level statistics of the receiver trace workloads are shown in Table 3.

One concern with our trace generation is that if the social networks are sparse and have very high average path length, the generated trace may not ensure locality of interaction while preserving the correlation between user degree and number of interactions. We cross check whether our traces preserve the intended key features. We do this by testing whether the newly generated traces preserve the two key features we aim to reproduce: (i) locality of interactions and

(ii) the correlation between user degree and number of profile views received. Both of these two features for the synthetic traces match quite closely (results not shown) with the original RR-PKU trace indicating that the synthetic workload generator retains the key properties of the original workload. We used 5 synthetic traces generated using different random seeds for each of the Facebook, Youtube and Flickr networks.

Crawler's attack topology We model crawlers by simulating that the crawler has compromised the accounts of random users in the social network. We simulate 1, 10, 100 and 1,000 corrupted user accounts in the RR-PKU, Facebook, YouTube and Flickr networks. As the crawler obtains access to more corrupted accounts in the network, he also acquires many more attack links to honest users. The varying strength of crawlers on different networks was discussed in Section 4.2 and Table 2.

Crawler's profile crawling trace To generate the crawler crawling workload, we follow the same crawler model discussed in Section 4.2. This models a crawler that achieves the lowest average path distance to the crawled profiles, and is the optimal attacker strategy.

Unless otherwise noted, all results are the average across 25 different runs of our simulator (5 synthetic honest user profile viewing traces, each paired with 5 synthetic crawler traces).

6.2 Trace-driven simulation methodology

To evaluate the performance of Genie, first we built a max-flow path based trace driven simulator. We use the social graph connecting the users to simulate a credit network. For each profile view in the workload trace, our simulator checks if there exists a set of paths in the credit network that allow $p-1$ units of credits to flow between the viewer and the viewee, where p denotes the shortest path length separating the viewer and the viewee. To this end, our simulator computes the max-flow paths [35] between the viewer and the viewee. If the max-flow is larger than $p-1$, then the profile view is allowed and if it is not, then the view is flagged. If the profile view is allowed, the credits along the links of the max-flow paths are updated as described in Section 5.

A key input to our simulator is the credit refreshment rate, which denotes the rate at which exhausted credits on the links are replenished. We set the credit refreshment rate in our simulator by tuning the following two parameters described in Section 5.2: (i) the initial credit value, i, assigned to each link in the network at the beginning of the simulation, and (ii) the credit rebalance rate, r_b, which restores some of the exhausted credits on the links after each time step, say of duration t. We set the parameter r_b to 1, which has the effect of restoring the credit values on all links to i after every refresh time period (2 weeks in our experiments). So $\frac{i}{t}$ represents the effective credit refreshment rate, which determines the number of profile views accepted both for crawlers and honest users. As the value of credit refreshment rate increases, more profile views will be accepted from both crawlers and honest users. Thus, the key evaluation challenge that we address using our simulator is: *Does there exist a credit replenishment rate that significantly slows down crawlers, while flagging few views by honest users?*

For a real-life deployment the OSN operator can estimate initial credits by using past browsing activity of users. The operator can build a distribution of user activity based on the imbalance between average number of profile views requested and received (weighted by the cost function mentioned in Section 5) made per outgoing link per user. Then the operator can pick an initial credit value/link from this distribution so that most (e.g. 99.9%) users' activity is allowed, but a few profile views are flagged (probably from super active users or crawlers). We check this methodology with our current RR-PKU dataset. Our implicit assumption here is that honest user behavior shows constant trends over time. We show the weighted imbalance distribution in Figure 6. From this figure we can estimate the number of users affected at a given credit value. For example with a credit value of 12, our estimate shows that views from 33,068 (99.3%) users will be allowed and 226 users will have some views flagged. We will show in section 6.4 that our estimate is quite good and indeed 275 users are flagged with this particular credit setting.

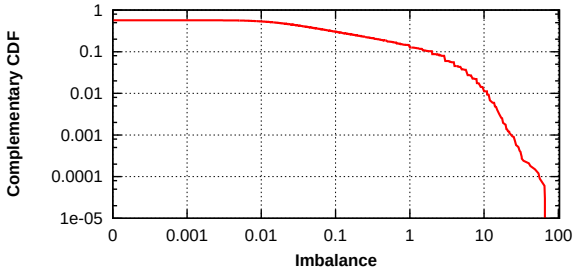

Figure 6: Complementary cumulative distribution of the imbalance of RR-PKU users (views are weighted by cost function mentioned in Section 5). Only 226 (0.7%) users have an imbalance greater than 12.

Scaling simulations to large graphs While we were able to run our max-flow path based simulator over the smaller RR-PKU network with 33,000 users, we found it computationally expensive to scale our simulations to the much larger YouTube and Flickr social networks with millions of users, links, and profile views. The computational complexity arises for three reasons: (i) even a single max-flow computation over a large graph is expensive, the most efficient algorithms for the maximum flow problem run in $O(V^3)$ [17] or $O(V^2 \log(E))$ [13] time; (ii) we have to perform millions of such computations, one for each profile view in the trace, and (iii) even worse, the computations cannot be parallelized and have to be performed online and in sequence, as the max-flow computation for a profile view has to account for credit changes on links in the network due to all prior profile views in the workload trace.

To allow Genie to be deployed on much larger networks, we leverage the recently proposed *Canal* [49] framework. Canal speeds up computations over credit networks and enables credit operations on very large-scale credit networks (on the order of millions of users) with low latency (on the order of a few milliseconds or less). Canal uses a novel landmark routing based technique to pre-compute paths with available credit (between different users in the network) continuously in the background as new credit operations are processed. Canal trades off accuracy for speed to achieve

Network	Avg. time (ms)	95^{th} percentile time (ms)
RR-PKU [22]	0.16	0.78
Facebook [47]	0.21	0.86
Youtube [27]	0.46	1.45
Flickr [26]	0.65	1.41

Table 4: Average and 95^{th} percentile time taken by Canal implementation to process one view request.

this goal. It explores only a subset of all possible paths between two users to complete a credit network operation between them. Thus, Canal may not always find sufficient paths with available credit between two users, even if such a path exists. However, Canal can achieve over 94% accuracy on various large-scale social networks [49]. Using Canal we were able to use Genie on data from online social networks including YouTube and Flickr that contain millions of users and links. We show the latency for processing one profile view with the Canal implementation of Genie in Table 4. Our current Canal implementation runs on a single machine. Results in [49] shows that, using a single machine with 48 GB RAM and 24 CPU cores, Canal can support operations on graphs with over 220 million links. Canal could likely be scaled to networks with a billion links using multiple machines and graph parallel processing frameworks [19, 25].

Simulating Genie with Canal We implemented Genie using the Canal library. While processing a profile view, Genie asks Canal for the shortest path length between the viewer and viewee. It should be noted that Canal can only provide an approximate shortest path length because it uses a subset of all possible paths between two users. Genie uses this path length as the basis of charging for the view and then queries Canal again for a set of paths from viewer to viewee with sufficient credit values. If Canal returns a set of paths then the view is allowed and Genie deducts credit along the returned set of paths. Otherwise the view is flagged. Canal requires two parameter settings to configure the amount of path data to be pre-computed. We use the same settings[4] used in the original Canal paper [49]. (These settings provided over 94% accuracy when applied to the Bazaar [35] system.)

As Canal may fail to find sufficient credit when it exists, but will not find credit that does not actually exist, deploying Genie with Canal provides an upper bound for the number of flagged profile views. We now examine how close the Canal estimates are to the true level of flagged user activity.

To understand the effect of approximation error introduced by Canal in terms of flagged honest user activities, we compare the Canal implementation output with the output from the max-flow path based technique. We use the RR-PKU network for this part of the evaluation as the relatively smaller size of RR-PKU enabled us to use the max-flow path based technique. Figure 7 shows the percentage of flagged honest user activity for the two techniques in the presence of a crawler with 10 compromised accounts. As the amount of credit available per refresh period increases, the percentage of flagged activity decreases for both techniques. On average, the absolute difference in flagged activities between the two techniques is only 0.7% of all user activities,

[4]20 level-3 landmark universes

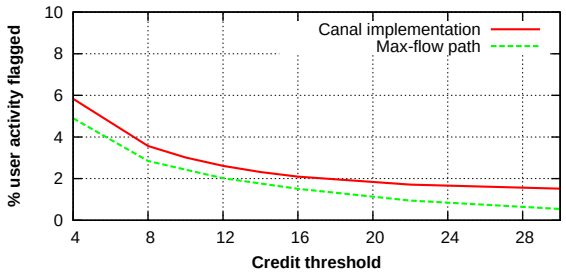

Figure 7: Variation of the fraction of user activities flagged with different credit values in RR-PKU in the presence of a crawler with 10 compromised accounts. We compare the results obtained using Canal implementation (red solid line) with those obtained using the max-flow based technique (green dotted line). The red solid line provides a close upper bound for the green line.

suggesting that Canal provides good accuracy. In the rest of our evaluation we will present results using our Canal implementation.

6.3 Limiting crawlers vs. flagging activity

We now switch our attention to the core tradeoff that is being made as we select the appropriate credit refreshment rate: namely, the amount of time it takes the crawler to crawl the entire graph and the fraction of honest users' activity that is flagged. We have already observed that to slow down crawlers effectively we need to replenish credits at a slow rate. However, a limited rate of credit replenishment opens up the possibility of honest users' views getting flagged. In this section, we explore the extent to which honest users' activity is flagged by Genie as it tries to limit crawlers.

We ran our Canal implementation for various different values of available credit per refresh period. For each replenishment rate, we compute two metrics: (i) the time it would take for a crawler to finish its crawl and (ii) the percentage of honest users' activity that is flagged. We compare these two metrics looking for a good tradeoff, where crawlers are effectively slowed down, while good user activity is rarely flagged. We present the basic tradeoff for our different social networks in Figure 8.

For each social network, we show the results for crawlers of different strengths. On YouTube and Flickr graphs with more than 1 million users, we considered a crawler controlling up to 1,000 user accounts, while for RR-PKU with only 30,000 user accounts, we limited the crawler strength to 10 users. While the absolute number of compromised accounts controlled by the crawler might seem small, it is worth noting that the percentage of compromised users in these networks is still substantial as discussed earlier in section 4.2.

The plots show that it is possible to slow down crawls sufficiently to force a crawler to spend several months to tens of months to complete a single crawl. At the same time, the percentage of flagged user activity can be held to less than 5%. In many instances, the flagged activity can be held lower than 1%. Thus, there are two important take-aways from these results: first, with Genie, a certain amount of honest users' activity will unavoidably be flagged. Second,

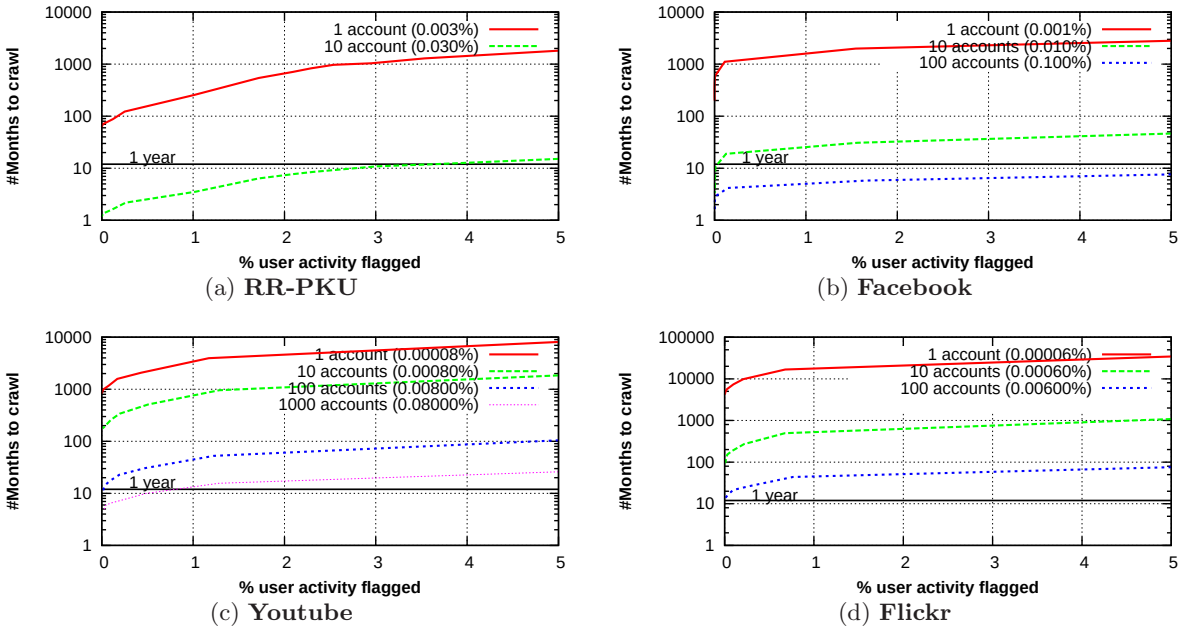

Figure 8: Trade-off between fraction of user activity flagged and time taken to finish a complete crawl with crawlers of varying strengths over different social network graphs. We measure crawler strength by the total number as well as the percentage (shown in parentheses) of compromised accounts in the network under control of the crawler. We conservatively allowed crawlers to exhaust the credits on links before allowing any honest users' activity.

unless the crawler is powerful and possesses over 0.1% of the accounts, the impact of the crawler on honest users is small.

6.4 Alternate strategies for flagged users

We observed in the previous section that a certain amount of blockage of honest users' activity is unavoidable. In section 5.2, we discussed that the OSN operator can make a choice based on some policy, once the profile view is flagged (or blocked) by Genie. Normally, the OSN operator would deny or delay the view to slow down the crawler's activity.

We now pose a simple question: can users do anything to minimize the amount of their flagged activity? To answer this question, we first investigate the flagged views in more detail. We then propose some recourse available to users with flagged activity.

We analyze the set of flagged activities in our extensive RR-PKU simulation, where we compute max-flow paths to verify if a profile view has to be allowed. We intentionally focused on max-flow based simulations because of the certainty that profile views flagged during such simulations are flagged due to lack of credit in the network.

For the analysis in this section, we focus on one particular simulation experiment with credit value 12, where 2.6% (or 2,574 activities) of the user activities are flagged and the crawler controlling 10 compromised accounts needs 8 months to complete the crawl.

A profile view can be flagged for one of three reasons: (i) the profile viewer runs out of credit on all links connected to itself (i.e., source blocked), (ii) the credit on links connecting the profile viewee is exhausted (i.e., destination blocked), or (iii) the view is flagged due to credit exhaustion somewhere in the middle of the network. Strikingly, only 190 out of 2,574 (7%) flagged views (i.e., 0.18% of all views) are flagged due to lack of credit on links in the middle of the network. The remaining 2,384 (out of which 1,961 views were blocked at the source) views which includes 93% of the flagged activities is due to credit exhaustion on links directly next to the viewers or the viewees. On examining the degrees of these viewers and viewees who are flagged, we find that 96% of them have degree 1 and 99% of them have degree of 5 or less. That is, most activities flagged near the source or destination is due to source or destination users having too few friends and lying on the fringes of the network graph. These results support our observation in Section 5 that social network graphs are sufficiently well connected in their core that most flagged activity (and credit exhaustion) occurs close to the fringes.

Next, we investigated the amount of flagged activities for individual users. We found that a small number of users are bearing the brunt of the flagged activities. 1,808 of the 2,574 (or 70%) of the flagged profile views are made by 3 users in the network. These are the same super-active users mentioned in Section 4.1. Interestingly, all 3 users issue two orders of magnitude more views than an average RR-PKU user and they are all flagged near the source. Further investigation suggests that these three users exhibit crawler-like characteristics with more than 60% of viewed profiles lying beyond their 2-hop neighborhood. Ignoring these three users (who bear strong resemblance to crawlers), the percentage of total flagged activity falls to less than third of its original value, which is already a low percentage (2.6%) of all activity.

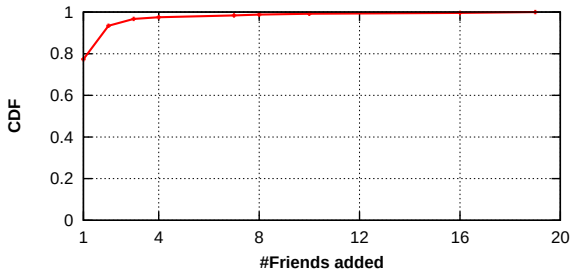

Figure 9: Cumulative distribution of how many extra links flagged RR-PKU users needed for completing their activities. Evidently majority of them needed just a few more links.

For the remaining users who contribute to only 30% of flagged views, we have already observed that most (99%) of the users have degree less than five. These are users who got flagged because their low number of friend links are insufficient to support the reasonable number (on average 6 views) of views they issued. However, we argue that there is a simple and natural recourse available to them: they can simply form more links in the online social network.

In order to test this hypothesis, we perform a simple experiment. We re-run the Genie simulation where each flagged user (i.e., 275 users falling in the low degree category), establishes a friend link to the destination of the flagged view (i.e requester sends a link request and the receiver approves it). This immediately leads to the acceptance of that view. At the end of the simulation, we look at the number of friend links established by each flagged user so that all the earlier flagged views could now be accepted.

Figure 9 shows the distribution of number of friend links established versus the ranked set of users. A significant majority (269 out of 275 or 97%) of the flagged users can get their views accepted by only establishing very few links (less than 4). Thus, most of the honest user activity flagged by Genie would be accepted if the users of the flagged views spent a minimal effort to establish a small number of friends. In fact, if Genie were to be deployed, it would naturally incentivize users to form a few more friend links. Given that many OSN sites already explicitly encourage their users to form more friend links, we believe that the overhead from Genie would be acceptable for a majority of users.

7. CONCLUSION

In this paper, we address the problem of preventing large-scale crawls in online social networks, and present Genie, a system that can be deployed by OSN operators to thwart crawlers. Based on trace data from the RenRen OSN, we show that the browsing patterns of honest users and crawlers are very different. While most honest users view the profiles of a modest number of users who tend to be nearby in the social network, even a strong, strategic crawler must view profiles of users who are further away. Genie exploits this fact by limiting the rate of profile views based on the connectivity and social network distance between a profile viewer and viewee. An experimental evaluation on multiple OSNs shows that Genie frustrates large-scale crawling while rarely impacting browsing of honest users; the few honest users who are affected can recover easily by adding a few additional friend links.

Acknowledgements

We thank our shepherd Athina Markopoulou and the anonymous reviewers for their helpful comments. We also thank Ben Zhao and Christo Wilson for their assistance with the RR-PKU trace. This research was supported by the Max Planck Society and NSF grants IIS-0964465 and CNS-1054233.

8. REFERENCES

[1] 45,000 Facebook accounts compromised: What to know. `http://bit.ly/TUY3i8`.

[2] Crawl packages for social networks. `http://80legs.com/crawl-packages-social-networks.html`.

[3] 83 million Facebook accounts are fakes and dupes. `http://bit.ly/Np3seb`.

[4] A standard for robot exclusion. `http://www.robotstxt.org/orig.html`, 1994.

[5] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha: Using hard AI problems for security. In *Proceedings of the 22nd Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'03)*, 2003.

[6] L. Backstrom, E. Bakshy, J. M. Kleinberg, T. M. Lento, and I. Rosenn. Center of attention: How Facebook users allocate attention across friends. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM'11)*, 2011.

[7] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida. Characterizing user behavior in online social networks. In *Proceedings of the 9th ACM/USENIX Internet Measurement Conference (IMC'09)*, 2009.

[8] C. Canali, M. Colajanni, and R. Lancellotti. Data acquisition in social networks: Issues and proposals. In *Proceedings of the International Workshop on Services and Open Sources (SOS'11)*, 2011.

[9] M. Cha, H. Haddadi, F. Benevenuto, and K. P. Gummadi. Measuring user influence in twitter:The million follower fallacy. In *Proceedings of the 4TH International AAAI Conference on Weblogs and Social Media(ICWSM'10)*, 2010.

[10] P. Dandekar, A. Goel, R. Govindan, and I. Post. Liquidity in credit networks: A little trust goes a long way. In *Proceedings of the 12th ACM Conference on Electronic Commerce (EC'11)*, 2011.

[11] G. Danezis and P. Mittal. SybilInfer: Detecting Sybil nodes using social networks. In *Proceedings of the 16th Network and Distributed System Security Symposium (NDSS'09)*, 2009.

[12] Details of 100 million Facebook users published online. `http://on.msnbc.com/qvLkX2`.

[13] E. A. Dinic. An algorithm for the solution of the max-flow problem with the polynomial estimation. *Doklady Akademii Nauk SSSR*, 1970.

[14] D. do B. DeFigueiredo and E. T. Barr. Trustdavis: A non-exploitable online reputation system. In *Proceedings of the 7th IEEE International Conference on E-Commerce Technology (IEEE E-Commerce'05)*, 2005.

[15] A. Ghosh, M. Mahdian, D. M. Reeves, D. M. Pennock, and R. Fugger. Mechanism design on trust networks. In *Proceedings of the 3rd International Workshop on Internet and Network Economics (WINE'07)*, 2007.

[16] M. Gjoka, M. Kurant, C. Butts, and A. Markopoulou. Practical recommendations on crawling online social networks. In *Selected Areas in Communications, IEEE Journal on Measurement of Internet Topologies*, 2011.

[17] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. In *Proceedings of the 18th annual ACM Symposium on Theory of Computing (STOC'86)*, 1986.

[18] Google Plus rate limiting. `https://developers.google.com/console/help/#cappingusage`.

[19] D. Gregor and A. Lumsdaine. The parallel BGL: A generic library for distributed graph computations. In *Proceedings of the Parallel Object-Oriented Scientific Computing (POOSC)*, 2005.

[20] Hacker proves Facebook's public data Is public. `http://tcrn.ch/9JvvmU`.

[21] Inside a Facebook botnet. `http://bit.ly/JSeRYs`.

[22] J. Jiang, C. Wilson, X. Wang, P. Huang, W. Sha, Y. Dai, and B. Y. Zhao. Understanding latent interactions in online social networks. In *Proceedings of the 10th ACM/USENIX Internet Measurement Conference (IMC'10)*, 2010.

[23] E. A. Kolek and D. Saunders. Online disclosure: An empirical examination of undergraduate Facebook profiles. *Journal of Student Affairs Research and Practice*, 2008.

[24] Y. Liu, K. P. Gummadi, B. Krishnamurthy, and A. Mislove. Analyzing Facebook privacy settings: User expectations vs. reality. In *Proceedings of the 11th ACM/USENIX Internet Measurement Conference (IMC'11)*, 2011.

[25] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: A system for large-scale graph processing. In *Proceedings of the International Conference on Management of Data (SIGMOD'10)*, 2010.

[26] A. Mislove, H. S. Koppula, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Growth of the Flickr social network. In *Proceedings of the 1st ACM SIGCOMM Workshop on Social Networks (WOSN'08)*, 2008.

[27] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM/USENIX Internet Measurement Conference (IMC'07)*, 2007.

[28] A. Mislove, A. Post, K. P. Gummadi, and P. Druschel. Ostra: Leveraging trust to thwart unwanted communication. In *Proceedings of the 5th Symposium on Networked Systems Design and Implementation (NSDI'08)*, 2008.

[29] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You are who you know: Inferring user profiles in online social networks. In *Proceedings of the 3rd ACM International Conference of Web Search and Data Mining (WSDM'10)*, 2010.

[30] A. Mohaisen, A. Yun, and Y. Kim. Measuring the mixing time of social graphs. In *Proceedings of the 10th ACM/USENIX Internet Measurement Conference (IMC'10)*, 2010.

[31] M. Mondal, B. Viswanath, A. Clement, P. Druschel, K. P. Gummadi, A. Mislove, and A. Post. Defending against large-scale crawls in online social networks. Technical Report 2011-006, MPI-SWS, November 2011. http://www.mpi-sws.org/tr/2011-006.pdf.

[32] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage. Re: Captchas: Understanding CAPTCHA-solving services in an economic context. In *Proceedings of the 19th USENIX conference on Security (SEC'10)*, 2010.

[33] M. Motoyama, D. McCoy, K. Levchenko, S. Savage, and G. M. Voelker. Dirty jobs: The role of freelance labor in web service abuse. In *Proceedings of the 20th USENIX conference on Security (SEC'11)*, 2011.

[34] Netflix-AOL data leak. `http://cnet.co/6JiHr8`.

[35] A. Post, V. Shah, and A. Mislove. Bazaar: Strengthening user reputations in online marketplaces. In *Proceedings of the 8th Symposium on Networked Systems Design and Implementation (NSDI'11)*, 2011.

[36] Public posting now the default on Facebook. `http://bit.ly/RkoIWR`.

[37] D. Quercia and S. Hailes. Sybil attacks against mobile users: Friends and foes to the rescue. In *Proceedings of the 29th Conference on Information Communications (INFOCOM'10)*, 2010.

[38] Rate limiting for yahoo! search web services. `http://developer.yahoo.com/search/rate.html`.

[39] Renren. `http://www.renren.com`.

[40] Spokeo privacy and safety concerns. `http://en.wikipedia.org/wiki/Spokeo#Privacy_and_safety_concerns`.

[41] T. Stein, E. Chen, and K. Mangla. Facebook immune system. In *Proceedings of the 4th Workshop on Social Network Systems (SNS'11)*, 2011.

[42] K. Strater and H. R. Lipford. Strategies and struggles with privacy in an online social networking community. In *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity (BCS-HCI'08)*, 2008.

[43] The day has come: Facebook pushes people to Go public. `http://rww.to/7Zhc6N`.

[44] N. Tran, J. Li, L. Subramanian, and S. S. Chow. Optimal Sybil-resilient node admission control. In *Proceedings of the 30th Conference on Information Communications (INFOCOM'11)*, 2011.

[45] N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-resilient online content voting. In *Proceedings of the 6th Symposium on Networked Systems Design and Implementation (NSDI'09)*, 2009.

[46] Twitter rate limiting. `https://dev.twitter.com/docs/rate-limiting-faq#measurement`.

[47] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in Facebook. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN'09)*, 2009.

[48] B. Viswanath, M. Mondal, A. Clement, P. Druschel, K. P. Gummadi, A. Mislove, and A. Post. Exploring the design space of social network-based Sybil defense. In *Proceedings of the 4th International Conference on Communication Systems and Network (COMSNETS'12)*, 2012.

[49] B. Viswanath, M. Mondal, K. P. Gummadi, A. Mislove, and A. Post. Canal: Scaling social network-based Sybil tolerance schemes. In *Proceedings of the 7th European Conference on Computer Systems (EuroSys'12)*, 2012.

[50] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove. An analysis of social network-based Sybil defenses. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'10)*, 2010.

[51] C. Wilson, B. Boe, A. Sala, K. P. N. Puttaswamy, and B. Y. Zhao. User interactions in social networks and their implications. In *Proceedings of the 4th European Conference on Computer Systems (EuroSys'09)*, 2009.

[52] C. Wilson, A. Sala, J. Bonneau, R. Zablit, and B. Y. Zhao. Don't tread on me: Moderating access to osn data with spikestrip. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Social Networks (WOSN'10)*, 2010.

[53] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A near-optimal social network defense against Sybil attacks. In *Proceedings of the IEEE Symposium on Security and Privacy (IEEE S&P'08)*, 2008.

[54] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. SybilGuard: Defending against Sybil attacks via social networks. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'06)*, 2006.