Iolaus: Securing Online Content Rating Systems

Arash Molavi Kakhki
Northeastern University
arash@ccs.neu.edu

Chloe Kliman-Silver
Brown University
chloe_kliman-silver@brown.edu

Alan Mislove
Northeastern University
amislove@ccs.neu.edu

ABSTRACT

Online content ratings services allow users to find and share content ranging from news articles (Digg) to videos (YouTube) to businesses (Yelp). Generally, these sites allow users to create accounts, declare friendships, upload and rate content, and locate new content by leveraging the aggregated ratings of others. These services are becoming increasingly popular; Yelp alone has over 33 million reviews. Unfortunately, this popularity is leading to increasing levels of malicious activity, including multiple identity (Sybil) attacks and the "buying" of ratings from users.

In this paper, we present Iolaus, a system that leverages the underlying social network of online content rating systems to defend against such attacks. Iolaus uses two novel techniques: (a) *weighing ratings* to defend against multiple identity attacks and (b) *relative ratings* to mitigate the effect of "bought" ratings. An evaluation of Iolaus using microbenchmarks, synthetic data, and real-world content rating data demonstrates that Iolaus is able to outperform existing approaches and serve as a practical defense against multiple-identity and rating-buying attacks.

Categories and Subject Descriptors

K.4.4 [**Computers and Society**]: Electronic Commerce—*Security*

Keywords

Content rating; Sybil; Vote buying; Social network

1. INTRODUCTION

Online content sharing services are a popular mechanism for users to find and share content; sites exist to share content such as business recommendations (e.g., Yelp, TripAdvisor), news articles (e.g., Digg, reddit), multimedia content (e.g., Flickr, YouTube), apps (e.g., iOS App Store, Google Play), and URLs (e.g., StumbleUpon, del.icio.us). Generally, these sites allow users to create accounts, declare friendships, and upload and rate content. The sites' extreme popularity is evidenced by the massive amounts of content that are uploaded: YouTube receives over 72 hours of new video uploaded every minute [35], and Yelp boasts reviews on over 889,000 businesses worldwide [47]. To locate relevant and trustworthy content from among this massive set of uploaded content, users are encouraged to rate content, with highly rated content receiving more prominent placement. The most highly rated content typically appears on the front page of the site or is listed more highly in search results, garnering significant attention and traffic.

Unfortunately, the increasing popularity of online content sharing sites has made them an attractive target for manipulation. For example, malicious users often attempt to ensure that their content is more highly ranked (or that others' content is more lowly ranked). On certain sites, such manipulation can have significant financial consequences: Recent studies have shown that increasing a business's overall rating on Yelp by one star can lead to 9% increase in revenue [19], explaining the numerous instances of rating manipulation that have been observed [1, 26, 30, 32, 33].

In general, manipulation on content rating sites is enabled by two separate attacks:

- Malicious users can create multiple identities (i.e., Sybils [10]), and use these identities to provide positive ratings on their own content or negative ratings on others' content [30, 32]. This is exacerbated by the fact that accounts are typically free to create, requiring only an email address and a solved CAPTCHA [42].
- Malicious users can "buy" positive or negative ratings from otherwise legitimate users by offering small compensation in exchange for ratings [1,33].[1] This is made worse by the fact that most content only receives a few ratings, making it possible to greatly influence the overall ranking with just a few additional ratings.

Such manipulation is undesirable for the site operator (whose reputation is negatively impacted by successful manipulation) as well as honest end users (who depend on the site to locate relevant and trustworthy content).

In this paper, we present the design and implementation of Iolaus,[2] a system that is designed to be run by the site operator to mitigate the effect of rating manipulation via the creation of multiple identities or the "buying" of ratings. Iolaus works using two techniques: *weighing ratings* and *relative ratings*. First, Iolaus leverages the structure of the

[1]This compensation can take multiple forms, depending on the particular site: for example, businesses can offer discounts for Yelp ratings [1], and users can offer reciprocal ratings for Flickr favorites [15].

[2]In Greek mythology, Iolaus was a nephew of Heracles. Iolaus provided essential aid to Heracles by helping to defeat the Hydra, a multi-headed monster who would grow two heads each time an existing head was cut off.

WWW 2013, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2035-1/13/05.

social network to bound the influence over the overall rating that malicious users can achieve via the creation of multiple identities. Iolaus assigns personalized *weights* to each rating, and selects the weights using a multi-commodity max flow formulation. Doing so ensures that the total weight of a single (human) user's ratings is bounded, regardless of the number of identities she creates.

Second, Iolaus uses the fact that most users provide few ratings to reduce the effectiveness of "buying" ratings. Instead of using a single rating directly as a raw score (e.g., content C gets ★★★), Iolaus transforms the user's rating to a ranking relative to all of the user's other ratings (e.g., C is in the top 10% of content). Since most legitimate users provide few ratings, "buying" ratings from random users provides significantly less benefits in Iolaus than it does today.

We demonstrate the effectiveness of Iolaus using three techniques. First, using microbenchmarks, we show that Iolaus has sufficiently low CPU and memory overhead to allow it to be practically deployed to the content sharing sites of today. Second, using synthetically generated simulation data and social network data from YouTube, we demonstrate that Iolaus performs as expected, strictly bounding the influence of malicious users and reducing their ability to "buy" ratings from random legitimate users. Third, we collect a complete dataset of businesses in two cities from Yelp, covering roughly 2M ratings on over 39K businesses provided by 1.5M users. We validate that Iolaus does not adversely affect the rankings for honest users in absence of malicious behavior, and is able to defend against multiple identity and purchased-rating attacks when applied to Yelp.

2. RELATED WORK

We now briefly cover related work, encompassing Sybil attack prevention and fake rating detection.

2.1 Blocking Sybils

Due to the attractive attack vector that free accounts provide, there is significant research interest in mitigating Sybil attacks. Traditional defenses against Sybil attacks rely on either trusted central authorities or tying identities to resources that are hard to obtain, such as social security numbers [5], mobile phones [24], or crypto-puzzles [2, 4, 6].

Recently, researchers have explored analyzing the structure of social networks as a mechanism for locating Sybil identities [9, 20, 29, 37, 44, 45] (a more extensive background is provided in [39]; we review the details relevant to Iolaus here). Unfortunately, there are two drawbacks to using existing Sybil defense schemes in content rating sites. First, existing schemes make the assumption that the honest region of the social network is densely connected with few internal small cuts [41] (formally, that the honest region is fast-mixing [25]). Recent work [18, 21] has cast doubt on this assumption, suggesting that existing Sybil detection schemes may end up accepting many Sybils or preventing honest users from interacting with each other [41]. Second, most pieces of content have few ratings; allowing even a small number of fake identities into the system can allow an attacker to "control" the rating for many items (for reference, SybilLimit [44] accepts $O(\log n)$ Sybils *per attack edge*).

2.2 Tolerating Sybils

Instead of trying to explicitly label identities as Sybil or non-Sybil, other approaches have focused on mitigating Sybil attacks in content rating services. Such systems are known as *Sybil tolerant* systems [39]. For example, DSybil [46] finds trusted users in the network (referred to as *guides*), and has provable optimality guarantees. However, DSybil can only provide recommendations for users who have submitted a sufficient number of ratings, which is often a small fraction of the population in practice. For example, in our Yelp data (fully described in Section 7), only 15% of users have provided more than 5 reviews. Also, DSybil is designed for rating systems where objects can only be either good or bad; Iolaus targets content rating systems that allow users to provide more fine-grained ratings.

SumUp [38] is another Sybil tolerant system that inspired our design. SumUp uses tokens passed over the social network in order to determine whether users' votes will be counted. While SumUp is conceptually similar to Iolaus, SumUp unfortunately has three weaknesses that Iolaus addresses: First, SumUp assumes that the region of the social network surrounding the user requesting the vote (called the *envelope*) is free of malicious users; if a malicious user is nearby, they receive many tokens and can issue many votes. Second, outside of the envelope, SumUp allows manipulation by malicious users: Honest users with multiple links are only allowed to place a single vote, while malicious users who divide their attack links across multiple accounts can potentially place multiple votes. Third, SumUp was not designed to address the "buying" of ratings from otherwise honest users. We demonstrate in Section 7 that Iolaus addresses these drawbacks and outperforms SumUp on real-world data.

2.3 Detecting fake ratings

Additionally, significant research has explored using data-mining techniques to detect and characterize rating manipulation. Systems have been built that use a variety of different inputs, including linguistic characteristics [27], user behavior [16,17,23], sets of recommended items [7], and common sets of user-reviewer pairs [43]. While these techniques can detect certain rating manipulation today, they rely on particular characteristics of malicious behavior. Regardless, such techniques could be used in combination with Iolaus.

3. IOLAUS DESIGN

We expect Iolaus to be deployed by the operator of a content rating site, such as Yelp or Flickr; we shall refer to this entity as the *operator*. Iolaus is designed to replace the existing content rating aggregation logic that the operator uses (i.e., instead of taking the average or performing review filtering [49], the operator would instead query Iolaus).

We assume that the operator collects ratings by a set of user accounts (referred to as *identities*) on a set of content objects; a user providing a rating on a given piece of content is referred to as a *rater*. We assume that non-malicious users provide honest ratings, with the exception of a small fraction of "bought" ratings.

We assume that the operator also provides the ability for users to declare "friends" and that friendship requires the approval of both parties; many content rating services (e.g., Yelp, Flickr, YouTube) already have such a social network. Similar to prior work [9, 37, 44, 45], we assume that links to a non-malicious user take effort to form and maintain. In other words, a malicious user cannot obtain an arbitrary number of links to non-malicious users. Note that we make

no assumptions about the difficulty of obtaining identities (a single person may have many identities), or the structure or links between the malicious identities. As a result, each (human) user has a cut in the network between identities that she owns and identities owned by other (human) users; while she can create identities and links on her side of the cut, she cannot unilaterally increase the size of her cut.

3.1 Input to Iolaus

We assume that the operator provides input to Iolaus:

- **Social network** Iolaus takes as input the list of social links between the identities. We assume this is represented as an undirected graph $G = (V, E)$, and that this graph is connected.
- **Ratings** Iolaus also takes as input the set of user ratings, represented by $(identity, content, rating)$ tuples. *identity* represents the user identity, *content* represents the content being rated, and *rating* represents the identity's rating.

3.2 System overview

The goal of Iolaus is to aggregate the ratings placed on content while ensuring that malicious users gain little additional influence by creating multiple identities or "buying" ratings. We make three observations that motivate Iolaus's design:

1. **Personalized aggregation** Most existing content rating aggregate schemes provide a single, global, aggregated rating for each piece of content (e.g., a business is ★★★ on Yelp). We take an alternate approach, allowing a personalized aggregated rating for each identity. Such an approach naturally captures legitimate differences of opinion (content ratings are, after all, opinions), and certain sites already provide personalized content ratings (e.g., Digg [36], Netflix [3]). We refer to the identity for whom we are calculating the aggregate rating as the *collector*.
2. **Weighing ratings** Existing approaches generally make a binary choice to either accept or reject each identity's rating when aggregating (e.g., Yelp's distinction between filtered and unfiltered reviews, SumUp's allowing or denying of votes). Instead, we *weigh* each identity's rating, and allow different identities to have different weights.
3. **Relative ratings** Finally, existing approaches view ratings as absolute (e.g., content C gets ★★★). Given that most identities rate few objects, this approach does not consider the amount of information each rater has provided (e.g., an identity who has only rated a single piece of content "counts" the same as an identity who has rated hundreds). In Iolaus, we transform raw ratings into relative ratings before aggregation.

In the following two sections, we describe how Iolaus chooses to weigh and interpret ratings, enabling it to defend against Sybil attacks and the "buying" of ratings.

4. MITIGATING SYBIL ATTACKS

Iolaus defends against multiple identity (Sybil) attacks through the *weighing* of ratings. Consider the set of raters $R \subseteq V$ on a single content object. Instead of taking the average of all ratings to be the aggregate rating, Iolaus uses a *weighting function* $w(r) \to (0, \infty)$ that assigns a positive weight to every rater $r \in R$. The aggregated rating is then simply the weighted average of these ratings

$$\frac{\sum_{r \in R} w(r) \cdot v_r}{\sum_{r \in R} w(r)}$$

where v_r is the rating of rater r. For existing systems which weigh all ratings equally, $w(r) = 1$ for all r.

The key challenge, then, is to select a weighting function that limits the ability for malicious users to gain additional aggregate weight through Sybil attacks (where a user's aggregate weight is the total weight of the subset of her identities in R). We also desire to select a *non-trivial* weighting function, which we define as a weighting function that assigns a non-zero weight to all identities.[3]

Below, we first formally define a Sybil attack in Iolaus's context, before detailing the properties we would like a weighting function to have. Finally, we describe the weighting function in Iolaus.

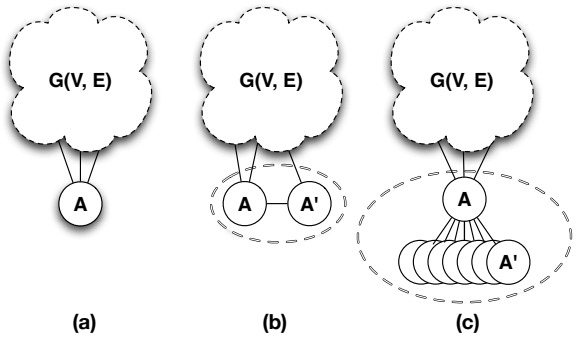

Figure 1: (a) A social graph, (b) malicious user A conducts a Sybil attack by splitting her identity, (c) malicious user A conducts a Sybil attack by creating a Sybil cluster.

4.1 Sybil attack

Formally, suppose that a malicious user controls a set of identities $I \subset V$. Consistent with prior work [45], we label the cut $(I, V \setminus I)$ as the *attack cut* (and the links along the cut as *attack links*), as these links signify links between the malicious user and identities controlled by other users. By our assumptions in Section 3, the number of attack links is bounded, but the number of identities in I and the number of links between these identities are unbounded.

As a result, a malicious user is able to perform three actions as part of a Sybil attack, depicted in Figure 1:

1. **Create additional identities** A malicious user is allowed to create any number of identities.
2. **Create links between owned identities** A malicious user is allowed to create links arbitrarily between identities she controls.

[3]A non-trivial weighting function is desired as we never actually know if a given rating was placed by a malicious or non-malicious identity; allowing a weighting function to give large numbers of identities 0 weight may discard most of the useful ratings.

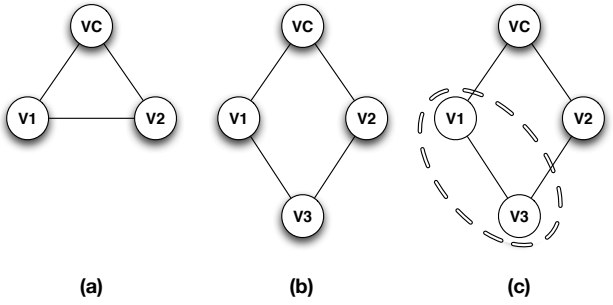

Figure 2: (a) a simple social graph consisting of three nodes a VC and two raters; V1 and V2, (b) same graph, but a new rater, V3, is added to the network, (c) same graph, but now V1 decides to split her identity by creating a new node, V3.

3. **"Split" identities on attack links** A malicious user is allowed to assign her end of her attack links to any of the identities she possesses. For example, if the malicious user possesses malicious identities $A_1 \ldots A_n$, and she has two attack links to non-controlled identities B and C, she can assign any of her A_i identities to be her endpoint of the attack links.

4.2 Sybil-proof

Ideally, we would like to select a weighting function that is *Sybil-proof*, meaning the weighting function ensures that a malicious user can gain no additional aggregate weight by conducting a Sybil attack. Formally, assume we have social networks G and G', where G' is the same as G except that malicious user A has conducted any number of Sybil attack actions (described in Section 4.1). For example, G and G' may be the graphs shown in Figure 1 (a) and Figure 1 (b) or (c). A Sybil-proof rating system would ensure that the aggregate weight assigned to raters that A controls is the same in both G and G'.

Unfortunately, Sybil-proof weighting functions on real-world networks are forced to be trivial, meaning they assign a weight of 0 to all raters that have multiple distinct paths to the collector (which, in practice, is almost all raters). To see why, consider the example shown in Figure 2 (b), where VC is the collector and $V1$, $V2$, and $V3$ are the raters. Consider rater $V3$, who has two distinct paths to VC. $V3$ could be (a) a legitimate, non-malicious identity, (b) part of a Sybil attack by rater $V1$, who splits her identity when linking to $V2$ as shown in Figure 2 (c), or (c) part of a Sybil attack by rater $V2$, who splits her identity when linking to $V1$. In either of the latter cases, each of $V1$ and $V2$ should get the same (aggregate) weight as in the network shown in Figure 2 (a). Thus, any weighting function that is Sybil-proof must assign $V3$ a weight of 0.

As a result, requiring a weighting function to be Sybil-proof precludes non-trivial weighting functions in practice. Instead, we must relax our requirements.

4.3 Sybil-bounded

We relax the requirement of our weighting function from being Sybil-proof to being *Sybil-bounded*. A Sybil-bounded weighting function is one where, given a social network G and malicious user A, there exists a bound $B_A > 0$ such that under any Sybil attack by A, the aggregate weight received by A's raters is always less than B_A. In other words, a malicious user may be able to get some additional weight through Sybil attacks, but there exists a bound on the total weight the malicious user will be able to receive, regardless of the number of identities A creates.

Compared to a Sybil-proof weighting function, a Sybil-bounded weighting function is strictly weaker (as malicious users can gain additional weight via Sybil attacks). However, we demonstrate below that (a) we can construct Sybil-bounded weighting functions that are non-trivial, and (b) we can select a weighting function that has tight bounds (leaving little additional weight to be gained via Sybil attacks).

4.4 Using max flow

Our goal now is to ensure that the weighting function is Sybil-bounded (i.e., that aggregate weight of the identities that the malicious user controls is bounded, regardless of how the malicious user conducts a Sybil attack). To do so, Iolaus expresses the problem of assigning weights as a multi-commodity max flow [11] problem,[4] viewing the social network as a graph with all links having unit capacity, and with the raters each sourcing a different flow, and with the collector serving as all flows' sink. We take the amount of flow that each rater is able to source as that rater's weight.

We choose multi-commodity max flow as it naturally has the Sybil-bounded property that we desire [34]. To see why, recall that the maximum flow between any two sets of nodes is defined by the minimum cut in the graph between the source and sink [13]. The attack links represent such a cut,[5] implying that the total flow—and therefore total weight—of the attacker is bounded, since the size of the attack cut is bounded by our assumptions in Section 3. Thus, regardless of how the malicious user conducts a Sybil attack, the aggregate weight of the attacker's ratings is bounded.

Moreover, using multi-commodity max flow also ensures that multiple malicious users gain no benefit from collusion. To see how, suppose that there are two malicious users. Without collusion, the two are each bounded by their respective set of attack links; should they collude, they are bounded by the union of their attack links.[6]

4.5 Ensuring tight bounds

Ensuring the presence of bounds limits the potential impact of a Sybil attack, but to be useful in practice, we would like to ensure the bounds are *tight*. Formally, we would like to minimize the difference between the assigned weights and the bound; doing so ensures that the malicious users can gain the least amount of weight by creating additional identities.

Ideally, we would like to solve the multi-commodity max flow problem using a linear solver such as CPLEX [8] or GLPK [14]. Unfortunately, expressing the problem *solely* as a linear maximization problem does not provide any guarantees about the tightness of the bounds to the assigned weights: to the linear solver, *any* solution which maximizes

[4]In brief, multi-commodity max flow maximizes the total aggregate flow between multiple source/sink pairs, with flows competing with each other for links' capacity.

[5]Of course, there may be additional, smaller, cuts between the attacker's identities and the sink.

[6]In fact, collusion may result in a lower aggregate bound, as the two users may have attack links to each other that become internal once they collude.

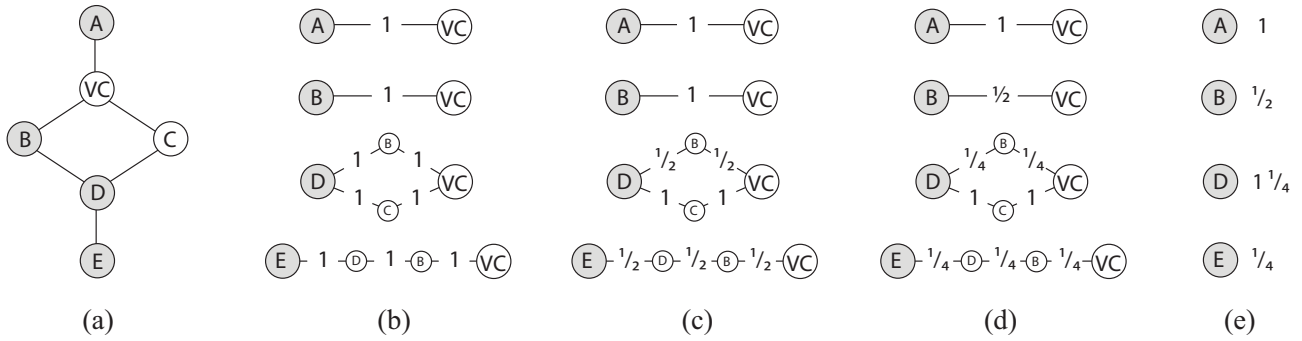

Figure 3: Example of determining weights in Iolaus. Shown are (a) the social network with collector VC and raters A, B, D and E (shaded), (b) paths selected for each rater, (c) resulting weights after normalizing $D-B$ link, (d) weights after then normalizing $B-VC$ link, and (e) final weights for each rater.

the total flow throughput is sufficient. As a result, such a solver may output solutions that have a very uneven distribution of the total weight across sources. For example, in the graph in Figure 2 (a), the solver may give $V2$ weight 2 (along its two paths to VC) while giving $V1$ weight 0.

Instead, Iolaus uses an approximation of max flow which results in more even distribution of capacity between raters and provides tight bounds on Sybil attacks, described below.

4.5.1 Determining weights

For each rater $r \in R$, Iolaus determines the max flow between r and the collector (ignoring all other raters), and derives a set of non-intersecting paths P_r for this flow. When multiple options for P_r exist, Iolaus selects arbitrarily between them. Next, Iolaus considers the graph where all raters attempt to send 1 unit of flow along each of their paths (i.e., for all r in R, each path in P_r is initially assigned weight 1). Since all links have unit capacity, there may be certain links that are over capacity (i.e., multiple raters have paths using that link). To resolve these situations, Iolaus *normalizes* the flow along these links by reducing the amount of flow proportionally. Formally, if a link is used by raters $\{r_1, r_2, ...r_n\}$ with weights $\{w_1, w_2, ...w_n\}$, each w_i is normalized with

$$w_i' = \frac{w_i}{\sum_{j=1}^{n} w_j}$$

where w_i' represents the value of w_i after normalization.

Iolaus normalizes links from the least-overcapacity to the most-overcapacity. Doing so ensures that malicious users are first bottlenecked at their attack cut before affecting any non-malicious users. To see why, recall that all of the paths begin at a rater but end at the single collector. Thus, the highest bottlenecks occur around the collector; links away from the rater are less likely to be over capacity.

4.5.2 Example

As an example of Iolaus's weighting in practice, consider the social network shown in Figure 3 (a), with collector VC and raters A, B, D, and E (C exists in the social network but has not given a rating). Iolaus first determines a set of non-intersecting paths with maximum flow for each rater; these are shown in Figure 3 (b). At this point, there are two links that are over-capacity: $B-VC$ has total weight 3 and $D-B$ has total weight 2. Thus, Iolaus first normalizes $D-B$ (reducing one of D's paths and E's single path to weight $\frac{1}{2}$); this is shown in Figure 3 (c). Then, Iolaus normalizes $B-VC$ (reducing B's path to $\frac{1}{2}$ and both of the previously reduced paths to $\frac{1}{4}$); this is shown in Figure 3 (d). At this point, all links are at or below capacity; the total weight of each rater's path is the weight for that rater (Figure 3 (e)).

4.5.3 Discussion

We observe that this weighting function provides Iolaus's Sybil resilience by providing a Sybil-bounded weighting system. For example, if node A in Figure 3 (a) were to attempt a Sybil attack, she would not be able to increase her aggregate weight beyond 1, regardless of the number of identities, or links between these identities, that she creates. She is always bounded by her attack cut of 1.

5. MITIGATING "BUYING" OF RATINGS

So far, we have described how Iolaus defends against multiple identity attacks; we now detail how Iolaus defends against the "buying" of ratings. Preventing the "buying" of ratings is one of the most difficult challenges facing content rating systems today: Since there is no cost for placing a rating, it is extremely hard to determine if a particular rating was indeed legitimate or the result of some form of out-of-band compensation. This challenge becomes worse by the fact that most content only receives a few ratings, making it possible to greatly influence the overall ranking with just a few additional ratings. The fact that being placed highly on certain content rating sites (e.g., Yelp, TripAdvisor) can have positive financial consequences [19] only further encourages compensation for positive ratings. To the best of our knowledge, no previous work has directly addressed the issue of dishonest ratings by legitimate users.

5.1 Relative ratings

Since placing a rating costs nothing to the identity, there is little to dis-incentivize the identity from doing so. In Iolaus, we add a virtual cost to placing a rating through the use of *relative ratings*. At a high level, relative ratings consider how content relates to other content the identity has rated.

To transform an identity's raw rating to a relative rating, we first consider all of that identity's ratings on other content objects. The relative rating is then simply the ranked score (between 0 and 1) relative to all of the identity's other ratings. For example, consider an identity who has provided

ratings ★★★★★ for content object c_1 and ★★ for content objects c_2 and c_3. We observe that the identity ranked c_1 higher than two other content objects; it is therefore in the "top third" of content objects for this identity. The relative rating for c_1 is therefore the midpoint of the top third of objects: 0.833 (the midpoint of [0.66,1]). Similarly, we observe that the identity ranked content objects c_2 and c_3 in the "bottom two-thirds" of content objects, but we do not know their order. Thus, the relative rating of c_2 and c_3 are both assigned to 0.333 (the midpoint of [0,0.66]).

Formally, suppose that the identity provided raw ratings $\{r_1, r_2, ...r_n\}$ on content objects $\{c_1, c_2, ...c_n\}$. For simplicity, assume these are sorted by r_i. Each content object c_i, then, is assigned the relative rating

$$\frac{i - 0.5}{n}$$

with the exception that any set of content objects that have the same raw rating are then assigned the average of all relative ratings with the same raw rating. Examples of converting raw to relative ratings for three different users are shown in Table 1. Note that all of user U_3's raw ratings are the same, so all end up with the same relative rating.

5.2 Discounting "bought" ratings

We now turn to examine how using relative ratings reduces the impact of "buying" of ratings. Consider an identity who is about to give a positive rating on a content object. Without relative ratings, the identity's positive rating would be viewed just like any other identity's rating, and there is no cost to the identity for providing the positive rating. In fact, without relative ratings, the identity could provide positive ratings on a number of content objects (i.e., "selling" her ratings multiple times) without any impact.

With relative ratings, the impact of "buying" ratings from a large number of identities is dramatically decreased. Since most identities provide few ratings (e.g., the average number of ratings per identity is less than two in our Yelp dataset), buying a positive rating from a random identity is unlikely to result in a high relative rating (as the n for most identities is quite small, meaning the resulting relative rating will be much lower than a similar rating placed by a legitimate identity who has placed more ratings).

Moreover, with relative ratings, placing a new rating causes some of the identity's other ratings to change, since inserting new rating into the ordered list of the identity's ratings affects the number of overall ratings (n), as well as the order of some of the ratings (i). Thus, providing a strongly positive rating of a new content object causes the relative view of other positive ratings to be lowered, ensuring that the identity can not simply repeatedly "sell" her ratings in exchange for compensation while providing the same benefit to all content items. Instead, to have the same effect as a single positive rating today, a user must simultaneously rate a large number of content objects negatively; this makes it much more difficult for malicious users to "buy" ratings from otherwise honest users.

		Rating		
User	**#**	**Raw**	**Transformed**	**Relative**
U_1	$r_{1,1}$	★★	0.25	0.25
	$r_{1,2}$	★★★★	0.75	0.75
U_2	$r_{2,1}$	★	0.1	0.1
	$r_{2,2}$	★★	0.3	0.3
	$r_{2,3}$	★★★	0.5	0.5
	$r_{2,4}$	★★★★★	0.7	0.8
	$r_{2,5}$	★★★★★	0.9	0.8
U_3	$r_{3,1}$	★★★★★	0.125	0.5
	$r_{3,2}$	★★★★★	0.375	0.5
	$r_{3,3}$	★★★★★	0.625	0.5
	$r_{3,4}$	★★★★★	0.875	0.5

Table 1: Example of converting raw ratings to relative ratings for three users. Raw ratings are first converted to transformed ratings; any transformed ratings with the same raw rating are then averaged.

6. DISCUSSION

In summary, the two parts of Iolaus work together to strengthen content rating sites. The use of max flow-based techniques ensures that users gain little benefit from creating multiple identities. The use of relative ratings reduces the effectiveness of "buying" positive ratings from random users, who generally do not have a significant rating history. We now discuss a few deployment issues with Iolaus.

Underlying network As discussed in the design section, Iolaus assumes existence of an underlying social network and will not be applicable to services that lack such network. Fortunately, most services today either directly have the social network or allow users to import their friends from other social networks such as Facebook and Twitter.

Disconnected users In Section 3, we noted that Iolaus assumes that the underlying social network is a connected graph. This assumption is not unique to Iolaus (all social network-based Sybil defense systems make a similar assumption [28,38,44,45]). In order to allow users who are not connected in the social network (e.g., guest users), Iolaus could be modified to create a "virtual" account for the user, with random links placed temporarily to allow rating calculation.

Rating interpretation Due to the use of relative ratings, the final ratings calculated by Iolaus will be a real-valued numbers between 0 and 1 (rather than, say, a number of stars). One potential concern is over how users will interpret such values. This range can trivially be mapped to any desirable range by a simple percentile conversion (e.g., the top 15% of content items receive ★★★★★).

Additionally, the ratings in Iolaus are personalized, meaning different users may see different rankings for the same content object. While this will clearly require an explanation to the users, existing sites such as NetFlix [3] and Digg [36] already provide personalized content ratings (implying that users do accept personalized ratings).

Impact on non-malicious users Another potential concern about using Iolaus is that it may change the current ranking of businesses, potentially for the worse. We evaluate this effect in Section 7, demonstrating that Iolaus does not adversely impact the rankings for non-malicious users.

7. EVALUATION

We now turn to evaluate the performance of Iolaus. We implemented Iolaus in C++ and Python. The implementation is divided into two parts: one that locates paths in the social network, and one that uses those paths and the rating history to calculate the rating.

Finding multiple, disjoint paths between nodes in a large social network is expensive, and naïve implementations can easily result in poor scalability. To avoid this poor scalability, Iolaus is implemented using Canal [40], a system that approximates credit payments in large credit networks. Canal uses landmark routing-based techniques to efficiently locate disjoint paths in large networks; we modified Canal to disable the credit transactions, and only use Canal to quickly find paths.[7]

The remainder of the Iolaus implementation consists of code that interacts with Canal, calculates weights, and transforms raw ratings into relative ratings. This part is implemented in 2,650 lines of Python.

7.1 Experimental setup

Social networks In the subsequent evaluation, we use both real-world social networks and synthetic social networks of varying sizes. Table 2 gives the statistics of the networks. The synthetic networks are generated using nearest neighbor method [31], with prescribed number of nodes, probability of adding new nodes, and number of random pairs connected. The resulting networks have been shown [31] to have characteristics close to real-world social networks.

The real-world social networks come from two large content rating sites: YouTube and Yelp. First, we use the social network of YouTube users [22], as originally used in the SumUp evaluation [38]. Unfortunately, the YouTube data set only contains the social network, and does not contain content ratings.

Second, we collect data from Yelp containing both social network information and content ratings from two cities: Boston and San Francisco. Specifically, we first determined the set of all businesses on Yelp located within the each city; this totaled 9,228 businesses in Boston and 30,339 in San Francisco. Then, we collected all ratings on these businesses; this totaled 278,719 ratings from 82,846 users in Boston and 1,655,385 ratings from 340,671 users in San Francisco.[8] Finally, we collected all of the social connections of these users; this resulted in a network of 383,557 users connected together with 888,335 links in Boston and 1,111,254 users and 3,920,553 links in San Francisco.

As Iolaus assumes that the social network is a connected graph, we only consider users located in the largest connected component (LCC) [22] of each Yelp graph. The LCC encompasses the vast majority of the data: In Boston, it covers 327,515 (85.3%) users connected by 883,179 (99.4%) links and providing 190,042 (68.1%) ratings. In San Francisco, it covers 1,303,086 (82.7%) users connected by 3,912,279 (99.8%) links and providing 1,303,086 (78.7%) ratings.

Simulating Sybil attacks Similar to prior studies [38], we simulate Sybil attacks by injecting malicious nodes and adding attack links (links from malicious nodes to non-malicious nodes). We refer to non-malicious nodes who are linked to by malicious users as *attacked nodes*. Inspired by prior work [41] we examine three different attack strategies for selecting attacked nodes:

Random Attacked nodes are chosen randomly.

k-closest Attacked nodes are chosen randomly among the k closest nodes (by hop distance) to the collector. This represents a targeted attack on a particular collector.

k-highest Attacked nodes are chosen randomly from among the k highest degree nodes in the network. This represents the most effective attack for being "close" for many collectors.

Note that we control the "power" of the attacker by varying k; a smaller k implies that the attacker can better target her attack (e.g., a small k in k-closest implies the attacker is able to obtain attack links very close to the collector).

Simulating "bought" ratings We also simulate the "buying" of ratings by malicious businesses in Yelp. To do so, we select random non-malicious users to provide "bought" ratings; each one of these users is simulated to provide one additional highly positive rating on the Yelp business that is trying to manipulate the ratings.

Comparing against SumUp and Yelp We compare the performance of Iolaus to SumUp [38] and a strawman version of Yelp's rating. For SumUp, we use the original code (obtained from the SumUp authors) and configured to the default values[9] prescribed in the original paper [38].

In practice, Yelp has a review filtering mechanism designed to block attacks, but its design is deliberately obfuscated [48]. As a result, we are unable to compare Iolaus directly against Yelp's filtering mechanism (as we do not know how it would perform when we introduce Sybil and rating-buying attacks).

Network	Nodes	Links	Average degree
YouTube	1.1 M	5.8 M	5.2
Yelp Boston	383 K	890 K	4.3
Yelp San Francisco	1.1 M	3.9 M	4.6
Synthetic 1	10 K	29 K	5.6
Synthetic 2	100 K	280 K	5.8
Synthetic 3	600 K	8.11 M	26.8
Synthetic 4	1 M	12.3 M	24.6

Table 2: Statistics of the social networks used for evaluating Iolaus. The synthetic networks are measurement-calibrated synthetic social networks [31].

7.2 Microbenchmarks

We begin by examining the amount of CPU time and memory required to determine an aggregate rating in Iolaus. Iolaus is designed to be parallelized; it can be configured to use multiple cores, and distributed across multiple machines, to

[7] In Iolaus, we configure Canal to use 15 2-level universes, three threads for creating universes, and sixteen threads for finding paths.

[8] Yelp divides reviews into unfiltered and filtered reviews; only unfiltered reviews are used by Yelp in determining a business' score. We collected both filtered and unfiltered reviews.

[9] We run SumUp with $\rho = 0.5$ and 20 non-greedy steps. However, since attackers can split identities on attack links (§ 4.1), the link pruning optimization in SumUp will only make it harder for honest users to find paths (Sybils can split their attack links across multiple identities, thereby avoiding the effects of pruning). Hence, we turn off this feature to make the comparison to SumUp fair.

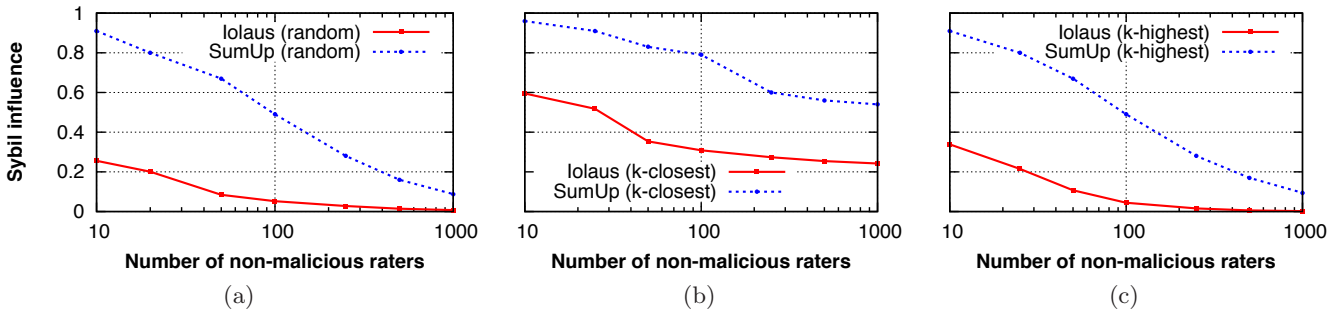

Figure 4: Sybil influence as the number of non-malicious raters is varied, for different attack strategies on the YouTube graph with 100 attack links. The graphs show (a) random attacked nodes, (b) k-closest attacked nodes, and (c) k-highest degree attacked nodes, with $k = 200$.

speed up computation time. We evaluate Iolaus deployed to a single machine with dual 8-core hyper-threaded Intel Xeon 2.1GHz CPUs and 128GB of memory.

Using the different networks, we select a single collector and a variable number of raters randomly from among all nodes. We then measure the time required to determine the aggregate rating, repeating the experiment 20 times and reporting the average. Figure 5 presents the results of this experiment. We observe that even when 100 users place a rating, the time required to determine the aggregate rating is under 5ms in all networks. In practice, most businesses would take substantially less: in our Yelp dataset, only 8% of businesses have more than 100 ratings. Moreover, the site operator could easily cache the calculated ratings, either with a fixed time-out or until a certain number of new ratings are provided.

In Iolaus, Canal stores the social network in memory. As a result, the memory requirements of Iolaus are determined by the memory requirements of Canal. On a similarly configured server to ours, Canal has been shown [40] to scale to networks containing hundreds of millions of links.

7.3 Comparison against SumUp

We now compare Iolaus directly against SumUp. As SumUp was only designed to mitigate the effect of Sybil attacks (and not rating-buying attacks), we only examine Sybil attacks here; in the following section, we examine both Sybil attacks and rating-buying attacks on our Yelp data set. We use the YouTube social network graph that was used in the original evaluation of SumUp [38].

While Iolaus is a weighing system which assigns a weight to every rater, SumUp either accepts or rejects a user's rating outright. Thus, directly comparing the two systems is not immediately straightforward. To make head-to-head comparison, we need a single performance measure which fits both systems. To do so, we consider SumUp also as a weighing system, which assigns weights 0 or 1 to raters that are rejected or accepted, respectively. We then define the metric *Sybil influence* as

$$\text{Sybil influence} = \frac{\text{Aggregate weight of Sybils}}{\text{Aggregate weight of all raters}}$$

representing the fraction of the total weight controlled by the Sybils. Thus, the smaller the Sybil influence value is, the better the system is at mitigating Sybil attacks.

7.3.1 *Varying non-malicious raters*

We first examine the effect of number of non-malicious raters on Sybil influence when the number of attack links is fixed. As the number of non-malicious raters increases, with a fixed number of attack links, we expect that both SumUp and Iolaus have lower Sybil influence. In this experiment, we select a random collector and 100 attack links, and vary the number of non-malicious raters. We repeat the experiment 20 times and report the average Sybil influence.

Figure 4 presents the results of this experiment when using the random, k-highest degree, and k-closest attack link strategies, with $k = 200$ (note that the k-closest scenario represents a very strong attacker, as there are over 82,000 total users). For all cases, Iolaus outperforms SumUp in reducing the impact of Sybils. The underlying reason is that in SumUp, for the random and k-highest degree attacked nodes, the Sybil raters are able to use each of the 100 attack links to get one rater accepted, allowing the Sybils to have significant influence.

In the case of the k-closest nodes strategy, Sybils are able to be part of SumUp's "envelope" around the collector, enabling them to cast multiple votes per attack link. With Iolaus, the Sybils' 100 attack links are forced to compete with all of the non-malicious raters' links. Sybils in Iolaus still manage to receive significant weight as they are very close to the collector, but have substantially lower influence than SumUp. With most content objects having few ratings, improved performance with few non-malicious raters is extremely important.

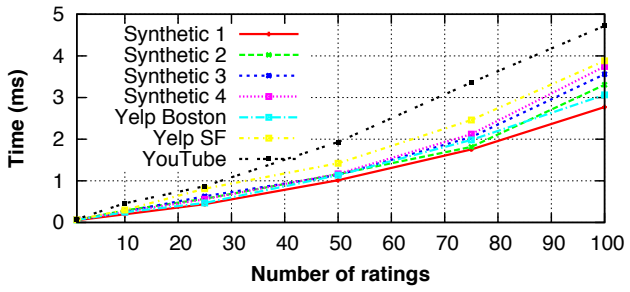

Figure 5: Average Iolaus running time for gathering up to 100 ratings in different networks.

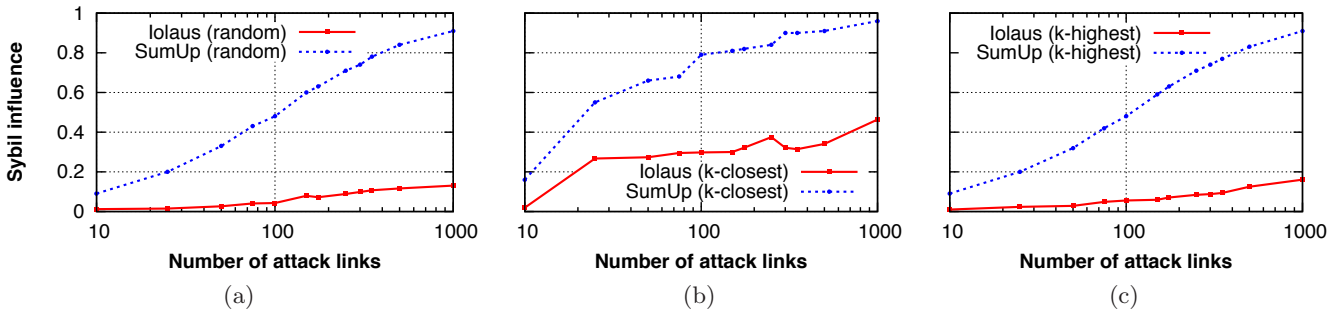

Figure 6: Sybil influence as the number of attack links is varied, for different attack strategies on the YouTube graph with 100 non-malicious raters. The graphs show (a) random attacked nodes, (b) k-closest attacked nodes, and (c) k-highest degree attacked nodes, with $k = 200$.

7.3.2 *Varying attack links*

We now examine the impact of the number of attack links on the resulting Sybil influence. We expect that as the number of attack links increases, the Sybil influence should increase linearly. In this experiment, we select a random collector, 100 random non-malicious raters, and vary the number of attack links. As before, we repeat the experiment 20 times and report the average.

Figure 6 presents the results of this experiment for all three attack strategies. We observe that Iolaus has lower Sybil influence than SumUp under all three cases. The reason for the superior performance is the same as before: in SumUp, the Sybil raters are able to use each of the attack links to get one rating accepted for random and k-highest attacks, and multiple ratings accepted for the k-closest attack. In Iolaus, the Sybil raters must compete with the aggregate links of the non-malicious raters.

7.4 Iolaus on Yelp

We now evaluate Iolaus on real-world data (including both social network and content ratings), examining Iolaus's resilience to Sybil attacks and rating-buying attacks. In this section, we use the Yelp data sets from Boston and San Francisco, described in Section 7.1.

7.4.1 *Ranking performance*

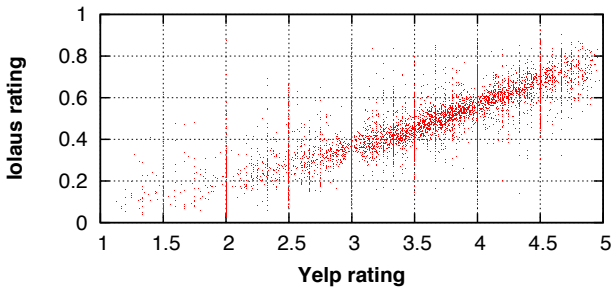

Figure 7: Scatterplot of Iolaus ratings versus Yelp's ratings for all Yelp Boston businesses. For the sake of accuracy, Yelp's ratings are not rounded to half-hops (as they typically are on Yelp's site). A strong agreement between the two ratings is observed.

We begin by examining the impact of using Iolaus on the overall ranking performance. In other words, how much does using Iolaus for aggregating ratings affect objects' rankings, even when Sybil and rating-buying attacks are not occurring? We use two approaches to address this question: First, we examine the global ranking of businesses; we compare these rankings to Yelp's current ranking. Second, we examine the per-collector ranking of businesses by comparing to ground-truth rankings provided by users.

In order to compare two rankings, we use the metric *Area under the Receiver Operating Characteristic (ROC) curve* or A'. In brief, this metric compares two ordered lists and represents the probability that the relative ranking of two items is in the same order in both lists [12]. Therefore, the A' metric takes on values between 0 and 1: A value of 0.5 represents no correlation between the lists, with higher values indicating a better match and 1 representing a perfect match between the two lists.

To examine the global ranking, we first rank all 9,228 Yelp Boston businesses using Iolaus for 10 randomly selected collectors. We then take the average of the Iolaus ranking across these 10 collectors to be the overall ranking of each business. Finally, we compare the order of ranked businesses in Iolaus to Yelp's order. We find that Iolaus's order compares to Yelp's order with an A' of 0.88. This indicates a strong agreement between the two orders, indicating that Iolaus does not significantly impact the ordering when Sybil and rating-buying attacks are not occurring. A scatterplot of the two rankings compared is presented in Figure 7.

Next, we compare the *ranking error* of Yelp, SumUp, and Iolaus. To do so, we first select a set of 500 users who have ranked at least 10 businesses. For each of these users, we calculate the Yelp, SumUp, and Iolaus rating of the businesses that user has rated, excluding the user's own rating. Each of these ratings are essentially *predicted* ratings; we then com-

City	Yelp–Filtered	SumUp	Iolaus
Boston	0.724	0.724	0.702
San Francisco	0.712	0.713	0.703

Table 3: Accuracy (A') of different systems in predicting users' rankings of businesses. All systems perform similarly, showing that Iolaus does not significantly altering the rankings of businesses in Yelp.

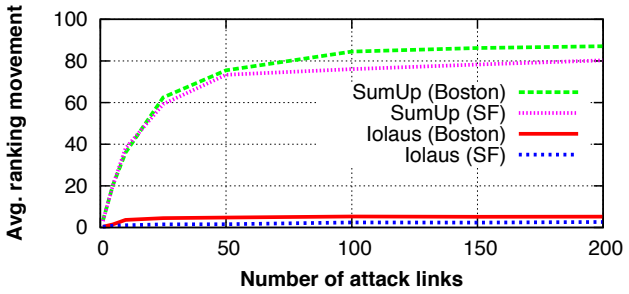

Figure 8: Average ranking movement of Iolaus and SumUp under different numbers of attack links.

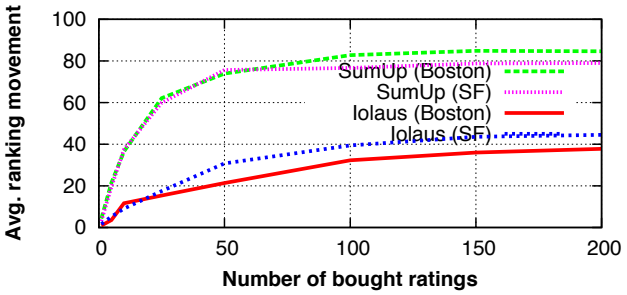

Figure 9: Average ranking movement of Iolaus and SumUp under different numbers of "bought" ratings.

pare the predicted ratings to the actual ratings provided by the user, and measure the differences using A'. Table 3 presents the results of this experiment for Yelp data in both Boston and San Francisco. We observe that all three systems are comparable, with Iolaus performing slightly worse. This indicates that Iolaus does not dramatically change the rankings of businesses.

7.4.2 *Defending against Sybils*

We now investigate Iolaus's performance on the Yelp dataset under Sybil attacks. For these experiments, we simulate Sybils placing highly positive ratings on a target business, and use the k-highest degree attack strategy with $k = 200$. Even though Figures 4 and 6 suggest that k-closest attack is the strongest, this attack is targeted at a particular collector. To influence the ranking for many collectors, Sybils are best served by attacking high-degree nodes.

Unfortunately, we cannot directly compare the SumUp's score with Iolaus's score (recall that SumUp's score is in terms of stars, whereas Iolaus's score is transformed to the unit interval). Thus, we compare the *relative* score of different businesses. To make the results comparable across cities and repetitions of the same experiment, in this section, we only consider businesses with exactly 10 ratings.

To measure the impact of the Sybil attack, we first select a *target business* from the lowest-ranked 25% of businesses with 10 ratings. The target business is the business that is trying to conduct a Sybil attack or "buy" ratings to increase its ranking. We then select a list of businesses to compare the target business against. We select these business with a wide distribution of ranks—up to 20 businesses with an average rating in each $\frac{1}{2}\bigstar$ interval—resulting in a list of 111 businesses (not all intervals contain 20 businesses). Finally, we measure the impact rating manipulation by measuring the difference (in terms of the number of places) the target business moves up in the ranking of 111 businesses after manipulation. We refer to this metric as *ranking movement*; lower ranking movement is better (an ideal system, of course, would allow 0 ranking movement).

Figure 8 shows the average ranking movement for 10 target businesses conducting Sybil attacks, averaged across 10 randomly selected collectors. With SumUp, the target business is able to significantly change the ranking, making itself appear much more highly ranked. This manipulation is possible as SumUp allows the Sybils to place an additional rating with each additional attack link. However, Iolaus manages to much more tightly bound the Sybils' influence, allowing significantly less ranking manipulation.

7.4.3 *Defending against rating "buying"*

Next, we investigate the ability for Iolaus to defend against rating-buying attacks. In these experiments, we do not add any Sybils or attack links, but instead, select a varying number of random non-malicious users to provide "bought" ratings. We simulate the non-malicious users providing highly positive reviews on a business. To evaluate the impact of this attack, we use the same businesses as in 7.4.2, and measure the impact of the attack using ranking movement.

Figure 9 presents the results of this experiment. As before, the results are the average across the same 10 collectors as in 7.4.2. We observe that, without any resistance to rating-buying attacks in SumUp, malicious users are able to greatly influence the overall ranking the business receives. However, with Iolaus, the overall impact on the target business's ranking is much lower, as the relative ratings reduce the impact of the purchased ratings.

Comparing Figures 8 and 9, we observe that rating-buying is a much stronger attack than Sybil attack, and has greater impact on final ratings. This result is expected, as bought ratings come from legitimate users who are likely well-integrated into the social network. However, we can see that Iolaus performs much better against such attacks in comparison to SumUp, which was not designed to protect against rating "buying."

8. CONCLUSION

We have presented Iolaus, a system that is designed to be deployed by the operator of an online content rating site and can defend against multiple-identity (Sybil) attacks and the "buying" of ratings. Iolaus is built using two techniques. First, Iolaus uses the weighing of different ratings to ensure that the total influence that any (human) user can have is bounded, regardless of the number of identities that the user creates. Second, Iolaus converts raw ratings into relative ratings, dramatically reducing the impact of the buying of ratings in practice. An evaluation demonstrated that Iolaus has low overhead and can be applied to the online content rating systems of today.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. We also thank Giorgos Zervas for his assistance with collecting the Yelp data. This research was supported by a Google Faculty Research Award, NSF grant IIS-0964465, and an Amazon Web Services in Education Grant.

9. REFERENCES

[1] K. Ashton. The Not So Secret Business of Fake Yelp Reviews. *Daily Deal Media*, 2012. http://www.dailydealmedia.com/657the-not-so-secret-business-of-fake-yelp-reviews/.

[2] M. Abadi, M. Burrows, M. Manasse, and T. Wobber. Moderately hard, memory-bound functions. *ACM ToIT*, 5(2), 2005.

[3] X. Amatriain and J. Basilico. Netflix Recommendations: Beyond the 5 stars (Part 1). http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html.

[4] N. Borisov. Computational Puzzles as Sybil Defenses. *IEEE P2P*, 2006.

[5] H. Chun, H. Kwak, Y.-H. Eom, Y.-Y. Ahn, S. Moon, and H. Jeong. Comparison of Online Social Relations in Terms of Volume vs. Interaction: A Case Study of Cyworld. *IMC*, 2008.

[6] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. *OSDI*, 2002.

[7] P.-A. Chirita, W. Nejdl, and C. Zamfir. Preventing shilling attacks in online recommender systems. *WIDM*, 2005.

[8] CPLEX. http://ibm.com/software/integration/optimization/cplex-optimizer.

[9] G. Danezis and P. Mittal. SybilInfer: Detecting Sybil Nodes using Social Networks. *NDSS*, 2009.

[10] J. R. Douceur. The Sybil Attack. *IPTPS*, 2002.

[11] S. Even, A. Itai, and A. Shamir. On the Complexity of Timetable and Multi-Commodity Flow Problems. *FOCS*, 1975.

[12] J. Fogarty, R. S. Baker, and S. E. Hudson. Case studies in the use of ROC curve analysis for sensor-based estimates in human computer interaction. *GI*, 2005.

[13] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Can. J. Math.*, 8, 1956.

[14] GLPK. http://www.gnu.org/software/glpk.

[15] L. H. Hwang, P. Damera, L. Brooking, and C. P. Lee. Promoting oneself on Flickr: Users' strategies and attitudes. *GROUP*, 2010.

[16] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. *WWW*, 2003.

[17] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw. Detecting Product Review Spammers using Rating Behaviors. *CIKM*, 2010.

[18] J. Leskovec, K. J. Lang, and M. W. Mahoney. Empirical Comparison of Algorithms for Network Community Detection. *WWW*, 2010.

[19] M. Luca. Reviews, Reputation, and Revenue: The Case of Yelp.com. *Harvard Business School Working Papers*, 2011. http://www.hbs.edu/research/pdf/12-016.pdf.

[20] C. Lesniewski-Laas and M. F. Kaashoek. Whānau: A Sybil-proof Distributed Hash Table. *NSDI*, 2010.

[21] A. Mohaisen and A. Y. A. Y. Kim. Measuring the mixing time of social graphs. *IMC*, 2010.

[22] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and Analysis of Online Social Networks. *IMC*, 2007.

[23] A. Mukherjee, B. Liu, and N. Glance. Spotting Fake Reviewer Groups in Consumer Reviews. *WWW*, 2012.

[24] G. Maganis, E. Shi, H. Chen, and D. Song. Opaak: Using Mobile Phones to Limit Anonymous Identities Online. *MobiSys*, 2012.

[25] M. Mitzenmacher and E. Upfal. *Probability and Computing*. Cambridge University Press, 2005.

[26] M. Moyer. Manipulation of the Crowd: How Trustworthy Are Online Ratings? *Scientific American*, volume 303, 2010. http://www.scientificamerican.com/article.cfm?id=manipulation-of-the-crowd.

[27] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock. Finding Deceptive Opinion Spam by Any Stretch of the Imagination. *ACL*, 2011.

[28] A. Post, V. Shah, and A. Mislove. Bazaar: Strengthening user reputations in online marketplaces. *NSDI*, 2011.

[29] D. Quercia and S. Hailes. Sybil Attacks Against Mobile Users: Friends and Foes to the Rescue. *INFOCOM*, 2010.

[30] M. Rose. Microsoft Investigating Claims of Ratings Manipulation in Xbox Live Indie Games. *Indie Games*, 2011. http://indiegames.com/2011/03/microsoft_investigating_claims.html.

[31] A. Sala, L. Cao, C. Wilson, R. Zablit, H. Zheng, and B. Y. Zhao. Measurement-calibrated graph models for social network experiments. *WWW*, 2010.

[32] D. Segal. A Rave, a Pan, or Just a Fake? *The New York Times*, 2011. http://www.nytimes.com/2011/05/22/your-money/22haggler.html.

[33] D. Streitfeld. Ferreting Out Fake Reviews Online. *The New York Times*, 2011. http://nytimes.com/2011/08/20/technology/finding-fake-reviews-online.html.

[34] S. Seuken and D. C. Parkes. On the Sybil-proofness of accounting mechanisms. *NetEcon*, 2011.

[35] Statistics – YouTube. http://www.youtube.com/yt/press/statistics.html.

[36] J. Turner. Personalization and the future of Digg. http://oreil.ly/brZbWK.

[37] N. Tran, J. Li, L. Subramanian, and S. S.M. Chow. Optimal Sybil-resilient node admission control. *INFOCOM*, 2011.

[38] N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-Resilient Online Content Voting. *NSDI*, 2009.

[39] 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. *COMSNETS*, 2012.

[40] B. Viswanath, M. Mondal, K. P. Gummadi, A. Mislove, and A. Post. Canal: Scaling social network-based Sybil tolerance schemes. *EuroSys*, 2012.

[41] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove. An Analysis of Social Network-based Sybil Defenses. *SIGCOMM*, 2010.

[42] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using Hard AI Problems for Security. *EuroCrypt*, 2003.

[43] G. Wang, S. Xie, B. Liu, and P. S. Yu. Review Graph based Online Store Review Spammer Detection. *ICDM*, 2011.

[44] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A Near-Optimal Social Network Defense Against Sybil Attacks. *IEEE S&P*, 2008.

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

[46] H. Yu, C. Shi, M. Kaminsky, P. B. Gibbons, and F. Xiao. DSybil: Optimal Sybil-Resistance for Recommendation Systems. *IEEE S&P*, 2009.

[47] Yelp 10-Q November 2012 Report. http://www.yelp-ir.com/phoenix.zhtml?c=250809&p=irol-sec.

[48] Yelp Which to Filter. http://www.yelp.com/faq#which_to_filter.

[49] Yelp's Review Filter Explained. http://officialblog.yelp.com/2010/03/yelp-review-filter-explained.html.