

Exploring the Impact of Profile Injection Attacks in Social Tagging Systems ^{*}

Maryam Ramezani, J.J. Sandvig, Runa Bhaumik, Tom Schimoler, Robin Burke,
Bamshad Mobasher

Center for Web Intelligence
School of Computing, DePaul University
Chicago, Illinois, USA

{mramezani,jsandvig,rbhaumik,tschimoler,rburke,mobasher}@cs.depaul.edu

Abstract. As in the case of all open and adaptive systems that rely on user input to organize and present content, social tagging systems are vulnerable to spamming and profile injection attacks. Malicious users may try to distort the system's behavior by inserting erroneous or misleading annotations, thus altering the way in which information is presented to legitimate users. Prior work on recommender systems has shown that studying the different attack types, their properties, and their impact, can help us find robust algorithms to make these systems more secure. In this paper we present and study two types of potential attacks against tagging systems. Using real data from a popular social tagging Web site, we empirically evaluate the impact of these attacks and their variants. Specifically, we consider two variants of an attack (called the overload attack) designed to promote a resource by adding different types of annotations to that resource, and another type of attack (called a piggyback attack) designed to promote a resource by associating it with other resources. We devise specific metrics to measure impact of these different attack types. Our results show that current systems are vulnerable to attacks, especially when the attack is focused on a specific target group of users to promote a target resource.

1 Introduction

Tagging systems are popular tools for organizing content; they allow users to annotate resources with one or more personalized tags. These social (or collaborative) tagging environments have gained popularity in part because they provide an open social environment for users to share resources and opinions without being hindered by pre-specified concept hierarchies or navigational structures. Social tagging systems are an extension of social recommendation behavior: people share their resource and tag preferences with one another, connecting them in an implicit social network. For example, in del.icio.us or Last.fm users can find other users with similar tags or resources and build a network with them.

Like other publicly accessible adaptive systems such as collaborative recommender systems, tagging systems present a security problem. Attackers, who cannot be readily

^{*} This work was supported in part by the National Science Foundation Cyber Trust program under Grant IIS-0430303.

distinguished from ordinary users, may inject biased profiles in an attempt to force the system to adapt in a manner advantageous to them. Recent work has established that adaptive Web applications, such as collaborative recommender systems, can be manipulated via “profile injection attacks”. In a profile injection attack (sometimes called “shilling”), an attacker uses fictitious identities to insert biased implicit or explicit ratings into a recommender system [1]. Such profiles may be generated manually by an attacker or an automated agent. These attacks do not require a great deal of knowledge about the details of the recommender system or its algorithms [2, 3]. In our previous works, we outlined the major issues in building secure recommender systems and proposed approaches to designing more robust recommenders and detecting attack profiles [4–6]

In this paper we broaden our view to other adaptive systems, specifically social tagging systems, and discuss their vulnerability in face of attacks designed to manipulate their output to users. Researchers have begun to study attacks against social tagging systems. Xu et al. [7] have introduced basic criteria for a good tagging system and proposed a collaborative algorithm for suggesting tags that meet these criteria. They have accounted for spam by assigning a reputation score to each user, based on the quality of the tags contributed by that user. Koutrika et al.[8] have proposed an ideal tagging system where malicious tags and malicious user behaviors are well defined. They propose a trusted moderator who periodically checks if user postings are “reasonable”. The moderator also identifies good and bad tags for any resource in the collection. The authors have also defined different strategies of attack, experimenting on the impact of different search algorithms. There is a set of good tags for each resource and if the user does not select from that set to annotate the resource, the tag is classified as spam. This approach, however, does not provide a distinction between a goal-oriented attack or a normal user who might be inclined to select a non-obvious tag for personal reasons. Krause et al. [9] use machine learning approaches to identify spammers in a social bookmarking system. They present features based on the topological, semantic and profile-based information and classify users as spammer or non-spammer. Heymann[10] et al surveys three categories of potential countermeasures against spam on the social Web based on detection, demotion, and prevention.

Prior work in this area, however, does not generally take into account the goals of an attacker, the audience of the attack, and the context in which the attack is mounted. We are interested in looking at this problem from a more practical perspective and try to determine what might motivate an attacker to spam the system. The goal of our research is to answer questions such as the following. What attack types are more successful? Which attack targets are more vulnerable? How many malicious users can a tagging system tolerate before results significantly degrade? How much effort and knowledge is needed by an attacker to attack the system? How can we detect attacks and protect the system? We defined dimensions of an attack in [11] including the intent and motivation of the attacker, the intended audience, the degree of profile obfuscation, and the size of the attack. We outlined a framework for navigating social tagging systems and based on the framework we introduced different types of attacks.

In this paper we describe two attack types in detail and study their impact on the system. We describe the *overload attack* in tag-resource context (in which a user navi-

	Starbucks	Coffeenatic	CoffeeExchange	Jonbucks
User1	coffee,café	coffee,espresso		
User2	coffee,food		coffee,mocha	
User3		coffee,blog		
User4	espresso	espresso		
User5		café	fairtrade,coffee	
User6	starbucks			
Attack1	café			coffee,starbucks
Attack2		blog	fairtrade	coffee,starbucks
Attack3	food			coffee,starbucks

Fig. 1. A hypothetical example of promoting a resource.

gates to resources by selecting a tag) and consider two variants for this attack: popular and focused attacks. The latter variant is designed to promote a resource to a specific group of users and is shown to be particularly effective. We also describe *piggyback attack* in a resource-resource context (in which a user selects a resource and is presented with other related resources). We explain how an attacker can actually implement these attacks and how the attacks will affect the system. We introduce the “Hit Probability” metric in order to measure the system-wide impact of an attack.

To see profile injection attacks in action against a tagging system, consider the example shown in Figure 1 of a tagging system that allows users to annotate URLs. A subset of tag assignments are displayed for users (User1 - User6). Suppose a user is searching for the resource that is most related to the tag “coffee”. Prior to attack, the system will display the resource *Starbucks* based on the number of occurrences of the tag. Now suppose another coffee shop, *Jonbucks*, wishes to promote the resource *Jonbucks* to users interested in coffee. Attack profiles (Attack1 - Attack3) are created, assigning the tags “coffee” and “starbucks” to *Jonbucks*. After the attack, the system now displays *Jonbucks* as the highest-ranking resource for the “coffee” tag while *Starbucks* will have a lower rank.

We give a brief description of attack types in section 2 and describe the details of our experiments and evaluations in 3.

2 Attacks Against Social Tagging Systems

The success of collaborative tagging is partially due to its facilitating the retrieval and discovery of resources within a single user-centric environment. Users browse the social tagging graph via the many associations between resources, tags, and users. Understanding the avenues for attacking a social tagging system requires analysis of this navigation process. However, there has been little formalization of tagging system output, and most research treats tagging systems solely as retrieval engines, ignoring the flexible browsing environment such sites offer. There is a need for a general model of navigation options and system outputs that can help us model the impact that an attacker may have.

A social tagging system consists of three generic elements: users, resources, and tags. Formally, the model can be described as a four-tuple $D = \langle U, R, T, A \rangle$, such

		<i>Target Element Type</i>		
		Resource	Tag	User
<i>Navigation Context</i>	Resource	Output: Related Resources Attack: Piggyback	Output: Common Tags Attack: Coattail	Output: Posing History Attack: Pivot Point
	Tag	Output: Popular Items Attack: Overload	Output: Related Tags Attack: Co-Occurrence	Output: Active Users Attack: Pivot Point
	User	Output: Recent Items Attack: Mole ("Shill User")	Output: Tag Cloud Attack: Mole ("Shill User")	

Fig. 2. Summary of Navigation Channels and Attack Types

that there exists a set of users, U ; a set of resources, R ; a set of tags, T ; and a set of annotations, A . Annotations are represented as a set of triples containing a user, tag and resource such that $A \subseteq \{\langle u, r, t \rangle : u \in U, r \in R, t \in T\}$.

The model can be viewed as a tripartite hypergraph $G = (V, E)$, where $V = U \cup R \cup T$ is the set of nodes and $E = \{\{u, r, t\} \mid \langle u, r, t \rangle \in A\}$ is the set of hyperedges [12]. To simplify analysis, we can reduce the hypergraph into three bipartite graphs with regular edges. These three graphs model the association between users and resources (UR), users and tags (UT), and tags and resources (TR) [13].

Bipartite graph views of the tagging system allow for aggregation of the associations between users, resources, and tags. Moreover, the views codify emerging patterns of organization within a tagging community and provide a means for social navigation. Each combination of element types R , U , and T represents a specific channel for presenting information in a tagging site, as shown in Figure 2.

A navigation channel has a particular context $r_c \in R$, $u_c \in U$, or $t_c \in T$ that refers to the current location of a user who is browsing or querying the system. A channel also defines a ranking algorithm for the elements $R_c \subseteq R$, $U_c \subseteq U$, or $T_c \subseteq T$ considered relevant to the context. For example, if a user selects the tag “coffee” (the tag context), the system will display resources that have been tagged with coffee, ranked by popularity or recency. It may also be possible to retrieve other tags related to coffee, or users who have employed the tag.

The success of an attack depends on generating the appropriate visibility for the attack target within the intended navigation channel. In many cases, the attack target is likely to be a resource, but could also be a tag or user. In addition, an attacker may focus on a particular navigation context as the reference point of attack. For example, the attacker in Figure 1 is trying to promote the Jonbucks resource within the tag-resource channel associated with the navigation context of the “coffee” tag. The attack may also have other impacts. It may for example make Jonbucks appear similar to Starbucks so that the resource is found via the resource-resource channel when Starbucks is the navigation context.

An attack against a social tagging system consists of one or more coordinated *attack profiles*. Each profile is associated with a fictitious user identity and contains annotations intended to bias the system. An *attack type* is an approach to constructing attack profiles, based on knowledge about the intended audience and the selected navigation channel.

A variety of attack types exist, each corresponding to a navigation channel as shown in Figure 2. We described the dimension of attacks and several different attack types in [11]. In this paper we examine several of these attacks in more detail and study their impact, empirically. Specifically, we examine the piggyback attack and two variants of the overload attack.

The goal of an *overload attack*, as the name implies, is to overload a tag context with a target resource so that the system correlates the tag and resource highly. The assumption is that the attacker wants to associate the target resource with some high-visibility tag, thereby increasing traffic to the target resource. In some cases, the attacker may have a more specific goal to promote the resource to some subset of users, likely buyers of his product, for example. The attacker would therefore be interested in associating the target resources with some specific tags likely to be used by prospective buyers and not be concerned with overall visibility for all users. He would therefore employ a *focused* version of the overload attack. We describe the details of these attacks along with examples in section 3.

The goal of a *piggyback attack* is for a target resource to ride the success of another resource. It exploits the idea of sharing tags among resources, attempting to associate the target resource with some resource context, such that they appear similar. The experiments below implement this idea by using tag duplication – picking a number of tags highly correlated to the resource context and annotating the target resource with the same tags.

3 Experiments

In this section we present preliminary results showing the impact of popular Overload, focused Overload and Piggyback attacks. For each attack type, we generate a number of attack profiles and insert them into the system database, testing the effects of different attack sizes and number of selected tag contexts.

3.1 Data Description

Our analysis is performed using data collected from the del.icio.us bookmarking service in which resources are URLs bookmarked by users. The data consists of the complete profiles of about 30,000 users of the service as of April 2007. The users were identified by crawling the site beginning at the most popular tag “design” and proceeding three levels deep into the user / tag link structure. The dataset contains 29,918 users; 6,403,441 unique URLs; 1,035,177 tags; 13,222,166 (User, URL) Pairs; and 47,185,789 (User, URL, Tag) Triples.

One of our goals is to determine whether tagging distribution of the target object influences attack effectiveness. It has been observed that the probability distribution of the number of users who tagged a URL follows a power law, in which a relatively small number of URLs are tagged with high frequency while all the rest occur with low frequency. The most popular URL has frequency of 14,353, while 99% of the URLs have frequency less than 77 and 50% of the URLs have frequency of less than 3. These numbers show that large portions of the data are in the long tail. Removing the long tail

means ignoring a huge part of the data, so we use all of the data in our experiments. However, since different parts of the distribution may show different behaviors, we divide the data into three partitions and run experiments on each partition independently.

We use the coefficient of variation (CV) to determine the partition boundaries as described in [14]. CV is a statistical measure of the dispersion of data points in a data series around the mean. It can be written as $CV = stdev/mean$ and is a useful statistic for comparing the degree of variation from one data series to another, even if the means are drastically different from each other. We partition the URLs into bins such that the collective CV of each bin is under 107, yielding three partitions. Partition 1 contains low frequency URLs tagged less than 1450 times, Partition 2 contains medium frequency URLs with a tag count between 1450 and 4850, and Partition 3 contains high frequent URLs tagged more than 4850 times.

3.2 Retrieval Algorithms

In tagging systems a user can click on a tag, resource, or a user and obtain a ranked list of related tags, resources or users. Within each navigation channel, a retrieval algorithm defines the particular elements considered relevant to the context. Relevance may be displayed in different ways between contexts, such as “popular tags”, “recent tags”, “recent resources”, “active users”, “related tags”, etc. Generally, results are based on popularity or recency, but there is no limitation. Some applications may also allow the viewer to choose the appropriate ranking algorithm. For example, del.icio.us allows a user to view the most popular or most recent resources that are annotated with the specified tag.

While research on retrieval models in tagging systems has introduced new algorithms in [15, 16], we focus on the vector space model [17] adapted from the information retrieval discipline to work with social tagging systems. We assume that retrieval is based on the Tag-Resource channel using the reduced TR bipartite graph; however, this framework may be easily modified to support retrieval in any navigation channel by using an appropriately defined bipartite graph.

A resource can be represented as a tag vector $r = [w_{t1}, w_{t2}, \dots, w_{tn}]$ such that w_t is the weight of a particular tag $t \in T$. Vector weights may be derived through many methods, including frequency or recency. In this work, we will rely on frequency. The *tag frequency*, tf , for a tag, $t \in T$, and a resource, $r \in R$ is the number of users who have annotated the resource using the tag. We define tf as:

$$tf(t,r) = |\{a = \langle u, r, t \rangle \in A : u \in U\}| \quad (1)$$

When a single tag is used as a query, as we assume here for simplicity, the resources are listed in order of their tf values since using $tf - idf$ will result in the same ranked list. For the resource-resource navigation channel, we represent a resource as a vector of tf values and use the cosine measure to compare vectors.

We consider here attacks that have the goal of improving the rank of the target item in some navigation channel. We can measure this value directly by looking at the rank of the target item before and after the attack. However, the average rank treats differences

at the top and bottom of the list identically. From the attacker’s point of view, a difference between a rank of 10 and a rank of 20 is far more significant than the difference between a rank of 110 and 120. For this reason, we measure the difference in reciprocal rank before and after an attack. Let r be the rank of the target item before the attack and r' be the rank afterwards. Rank improvement Imp is then given by $Imp = \frac{1}{r'} - \frac{1}{r}$. It is easy to see that this metric has the properties that we want. Imp is maximized (close to 1) when an item that is at the end of the list (maximum r) moves all the way to the front of the list ($r' = 1$). A item moving from 20th to 10th would get a score of 0.05, but one moving from 120th to 110th would only score 0.00075.

The Imp metric is local. It measures the impact of an attack on a single channel within a particular navigation context. We are also interested in global measures that can capture the overall impact of an attack. In an analogy to the PageRank algorithm, we approximate the probability that a user choosing tags randomly would encounter a given target resource. The first step is to estimate the probability that a user will click on a specific tag as the ratio of the frequency of that tag to the sum of the frequencies of all tags in the system. Therefore, we have a probability for each tag represented by $P(t_i)$ computed as follows:

$$P(t_i) = \frac{tf(t_i)}{\sum_{j=1}^N tf(t_j)} \quad (2)$$

where tf is the tag frequency.

We next find the average likelihood that the target resource is in the top n results of a tag context for a user navigating randomly through the system. We are interested therefore in the top n hit score where value is 1 if the target resource appears in top n results, and 0 otherwise. For each tag t_q we multiply $P(t_q)$ by the hit score. Thus hit probability for each resource is $P(t_q)$ if the resource appears in the top n list, and 0 otherwise. Hit probability for multiple tags is the sum of their individual hit probabilities. In our experiments, we use the average hit probability for all target resources in each partition of the data. More formally, the hit probability of resource r_i given top n results $R_t \subset R$ of a tag t_q is:

$$HitProb(r_i, t_q, n) = \begin{cases} P(t_q) & \text{If } r_i \in R_t \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Where R_t is the top n returned results for the query tag t_q .

Consider the attack shown in Figure 3. The number in each cell shows the tag frequency (tf) for each tag-resource pair. (Let us ignore for the moment the bottom line of the table “Focused overload”). The table shows the results of adding 20 attack profiles to the system, each containing the target URL and two popular tags (design and web). We can find the probability of each tag using the above-mentioned formula. If we assume this matrix shows the entire system, the sum of all tag frequencies will be 307 after the attacks. So, the probability for tag “design” will be the sum of frequencies of “design” over all urls divided by 307, which is 0.41. In the same way we can find the probability of tag “coffee” will be 0.088. Hit probability of each URL is equal to the

tag probability if they appear in the search result, zero otherwise. Let us assume a result list of size 3: $n = 3$. The Hit Probability for URL2 for tag “design” is 0.41 while the Hit probability for tag URL3 is zero since it does not appear in the top n results. Because this measure covers all of the tag-resource channels, it is a more global measure of an attack’s impact than the *Imp* metric. However, it still does not include all possible ways that a user could encounter a given resource. A holistic measure that includes all the navigation channels is a subject for future work.

3.3 Overload Attack

Resource-Tag matrix	Coffee	starbucks	café	design	web
URL1	5	3	2		
URL2				50	30
URL3	2	3	1	1	
URL4	10	20	5	5	
URL5				50	50
Target URL (Popular overload attack)				20	20
Target URL (Focused overload attack)	10	10	10		

Fig. 3. An example of the resource-tag matrix after an Overload attack

The goal of an overload attack is to associate a set of tags with a target resource so that the system retrieves the target resource when users search for the selected set of tags. This attack comes in two types: “popular” and “focused”. A popular attack is one that is aimed at all users, so the attacker will choose the most popular tags with which to associate his resource. A focused attack is aimed at a subset of users, perhaps defined as likely buyers, and the tags are chosen to be those that these users would employ.

Popular Overload Attack Figure 3 shows an example of the resource-tag matrix in which two sets of attack profiles have been added. The first attack row (in grey) shows the Popular Overload attack – the tags “design” and “web” are the ones used most frequently in our del.icio.us dataset. If we assume that the system returns 3 URLs to the search query, the returned URLs for query tag “design” before attack are URL2, URL5, and URL4. After the attack, the returned URLs will be URL2, URL5 and the target URL.

For our experiments, we use the 50 most frequently used (popular) tags from our database and we test the attack effectiveness in the three different partitions of resources, randomly selecting 50 resources from each partition, and averaging the results. We randomly select tags from the set of 50 popular tags and associate the target URL with the selected popular tags. We use the hit probability and rank improvement measures to evaluate the attack impact at different attack sizes. We measure “size of attack” as a percentage of the actual users in the system. There are approximately 29,000 users in the database, so an attack size of 1% corresponds to 290 attack profiles added to the system.

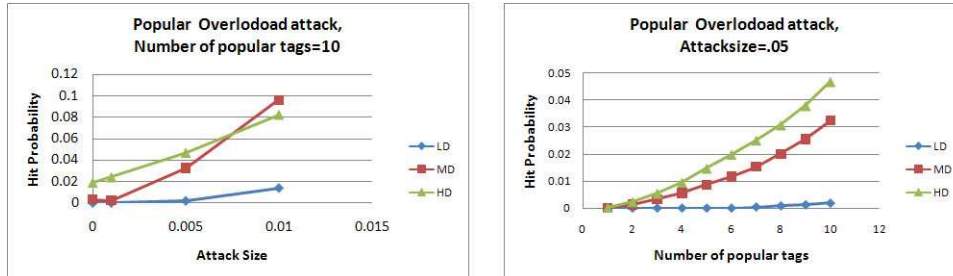


Fig. 4. Left chart shows Hit Probability for different attack sizes when the number of popular tags is 10 in each attack profile. Right chart shows Hit Probability for different numbers of selected popular tags at .005% attack size

Figure 4 depicts the effect of varying attack sizes with 10 popular tags associated with the target resource. 10 popular tags are randomly selected from the set of 50 most popular tags in del.icio.us. Hit probability values before an attack are very low for Partition 2 and Partition 3, and zero for Partition 1. This behavior is expected, as the chance that low frequency URLs show up in search results for popular tags is very small before an attack. After attack the hit probability increases steadily as the number of attack profiles increases for all three partitions. However, as we can see in figure 4 attacks do not have much effect on the low frequency URLs while the hit probability for the other two partitions increases significantly with increasing attack size. This behavior is due to the fact that hit probability is based on top-n hit ratio and it is much more difficult to push less frequently used URLs to the top-n list than to push more frequently used URLs, which already have much higher hit probabilities before the attack.

Figure 5 shows the rank improvement for URLs in different partitions for different attack sizes. We can see that all three partitions are affected similarly by increasing the attack size. Rank improvement is a local measure that shows the impact of the attack in searching a specific tag. Since rank improvement is not based on top-n result, we expect to see the same behavior for all three partitions. Even though the attack will not be successful to bring the low-frequency URLs to the top-20 results, it moves the rank of those URLs to a higher position in the list. We increase the attack size to determine the highest possible improvement and find that an attack size of 10% guarantees that the target URL will be the top result for the searched tag. In our data set this means an attacker needs to inject approximately 3000 fake profiles in order to get a 100% rank improvement.

Focused Overload Attack In a focused attack, the attacker is targeting a focused group of users, for example, those who are interested in coffee. Such an attack would contain profiles that associate the target URL with tags related to coffee. In Figure 3, we see a focused attack containing 10 attack profiles using the three tags “coffee”, “starbucks”, and “cafe” associated with a target URL. If a user searches for “coffee”, URL4, URL1, and URL3 will be returned before the attack but after the attack the results will be URL4, the target URL, and URL1. As we can see, in each type of overload attack

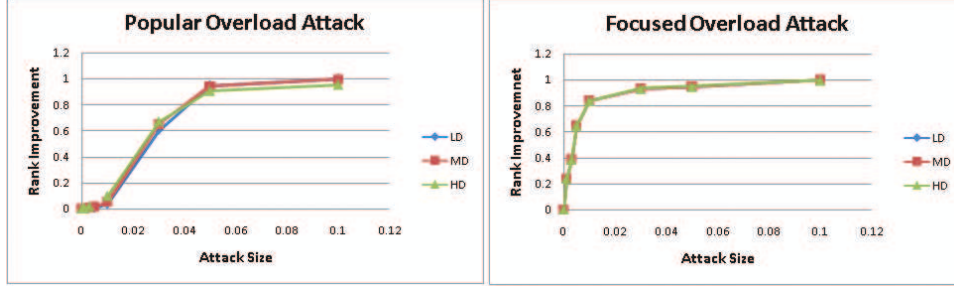


Fig. 5. Left chart shows rank improvement in popular overload attack and the right chart shows the rank improvement for focused overload attack

User-Tag matrix	Coffee	starbucks	café	design	web
User1	2	2	1		
User2				10	5
User3	2	3	1	1	1
User4		2	3	5	2
User5	3	4		1	3

Fig. 6. An example of the user-tag matrix

there are two factors which can change the attack effect: one is the attack size and the other is the number of popular or focused tags in each attack profile.

For the focused attack, we do not use the global hit probability because the attack is designed for a subset of users. We modify our metric to find the probability of the focused tags in the focused group. To this end, we first find the focused users and take into account only those users in computing the Hit Probability. More formally, we find a new tag frequency (tf_f) for each tag-resource pair where $tf_f(t, r)$ is the number of times resource r has been annotated with the tag t by the group of target users. We find *focused hit probability*, $P_f(t_i)$, which estimates the probability that a random user from the target group clicks on (or searches for) tag t_i .

Consider again the example in Figure 3. To find the focused hit probability, we need the user-tag matrix in addition to the resource-tag matrix to find the target users. Figure 6 shows a example of the user-tag matrix.

If the goal of the attack is to target a group of users who are interested in coffee, we should first find all related tags to the tag “coffee” and then find the users who have used those tags in their profiles. Suppose, in our example, the related tags to coffee are “starbucks” and “cafe”. From the user-tag matrix we can see that user1 and user3 have used all related tags so they are the focused group. The sum of all used tags in the focused group is the sum of all tags used by user1 and user3 which is $5+8=13$. Thus, $P_f(\text{coffee}) = 4/13$ and $P_f(\text{starbucks}) = 5/13$. We use these probabilities to find the Hit Probability of each URL as before.

For the focused attack experiments, we first find a set of related tags to a specific tag. We use the related tags from del.icio.us Web site. To find the focused group for

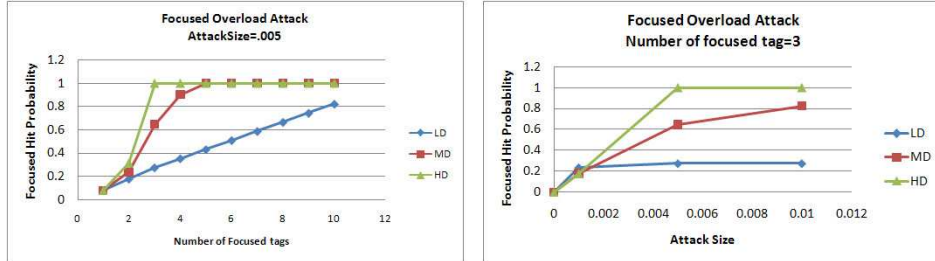


Fig. 7. Left chart shows Focused Hit Probability for different attack sizes when the number of focused tags is 3 in each attack profile. Right chart shows Focused Hit Probability for different number of selected focused tags at .005% attack size

those tags, we select the users who have at least five of the related tags in their profile. We calculate the focused hit probability to find the effect of the attack at different attack sizes and different numbers of focused tags selected from the related tag set. Figure 7 shows the results of the focused overload attack for different attack sizes and different numbers of focused tags in each attack profile. As we can see from the results, a comparison between popular and focused attacks shows that focused attacks have a significant impact on the system at lower attack sizes. For example, at .005 attack size, using more than 3 focused tags in the attack profiles will result in hit probability of 1 in the MD and HD URLs and about .5 hit probability in LD URLs while at the same attack size in popular overload attack the hit probability is limited to .05. The fact that the focused attack is designed to target a specific group of users makes it more successful with less effort. Even though the focused hit probability is calculated only for target users, this reflects the goal of the attack. We can see that using more focused tags in the attack profile can increase the focused hit probability across all partitions. We expect this result since we are adding up the hit probabilities from different tags. For example, if the target URL is associated with tags “coffee” and “starbucks” then it will have a higher hit probability because it will be returned to users who search for tag “coffee” and also to users who search for tag “starbucks”. The right side of figure 5 shows the rank improvement for the focused overload attack. We can see the same behavior as the popular attack across all partitions. It is interesting to see, however, that the curve has a sharper increase at low attack sizes in the focused attack. For example, at attack size 2% we can see that focused attack rank improvement is more than .8 while the popular attack rank improvement is about .2. This confirms our hypothesis that the focused attack is more effective than the popular attack for small attack sizes.

3.4 Piggyback Attack

The goal of a piggyback attack is to associate the target resource with other resources, such that they appear similar. For example, an attacker might identify a particular page as being highly popular among the site’s visitors and seek to make his target page appear in the “Related resources” navigation channel. We implemented the Piggyback attack using the tag duplication strategy described above by adding attack profiles to the sys-

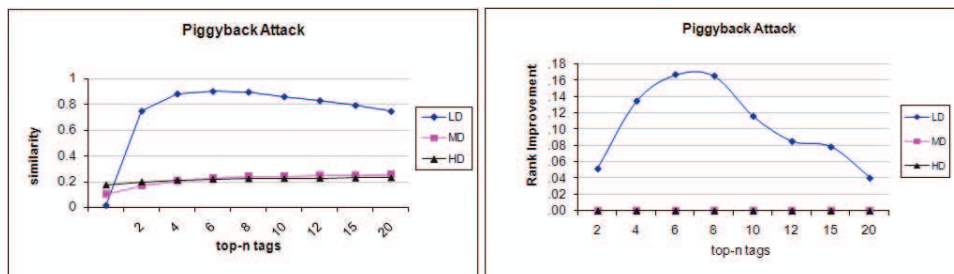


Fig. 8. The left chart shows similarity and right chart shows rank improvement of Piggyback attack between popular URLs and target URL for varying duplicated tag numbers at 17% attack size.

tem that associate the target resource with a set of tags which are the most frequently used tags for the selected popular resource.

We consider each resource as a vector of tags, which stores the frequency of users who have associated each tag to that resource. In addition to the rank improvement measure, for this attack, we also measure the cosine similarity between selected popular resources and the target resource. We have selected 5 popular resources and 10 target resources from each category. Our results include average similarity over selected popular resources and all target resources in each partition.

We look at the impact of attack by changing two variables: attack size and number of selected tags from popular resources that are associated to the target resource. Attack size is the number of users added to the system. We look at impact of the attacks for each different partition of the distribution.

The left chart of Figure 8 displays similarity when changing the number of duplicate tags. As the results show, the similarity between popular resources and low frequency URLs changes dramatically after the attack. The reason is that the number of tags associated with low frequency URLs are very small before attack. Adding 50 profiles that duplicate tags from selected popular resources will have a large effect in the similarity between the target resource and selected resource. However, similarity scores don't change much for other partitions before and after an attack, as these resources are already associated with many common tags prior to attack.

The right side of Figure 8 shows the average rank improvement between the target resource and popular resources. This result indicates that, for low-frequency URLs (Partition 1), as similarity increases the rank also increases. After adding the top 6 tags from popular resources, the rank of the target resource is at position 5 and results in a rank improvement of .16. For other partitions, similarity scores don't change significantly and there is no improvement in the ranking.

Figure 9 shows the results for similarity with varying attack size. It indicates that even a small number of attack profiles (.03%) added to the system with the 6 top tags selected from popular resources exposes vulnerability in low frequency URLs. The similarity score changes from .014 before attack to .85 after attack. As expected, the similarity scores for the other partitions do not change much before and after attack.

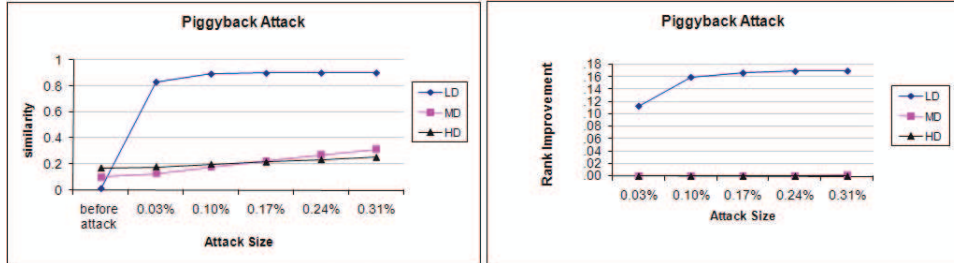


Fig. 9. The left chart shows similarity and right chart shows rank improvement of Piggyback attack between popular URL and target URL for varying attack size, with top 6 tags duplicated

Similar results can be observed regarding rank improvement. The results of our experiments show that low frequency URLs are vulnerable to piggyback attacks while high frequency URLs are more robust. This can be explained by considering the cosine similarity metric: for a low-frequency URL, the initial tag vector is very sparse and is therefore heavily altered by the attack; URLs from the MD or HD distributions already have many tags associated with them (their tag vectors are less sparse), so the effect of the focused attack is largely absorbed through vector normalization.

4 Conclusions and Future Work

In this paper, we have discussed the problem of security and robustness in social tagging systems in the context of several specific attack types. We modeled two attack types, Overload and Piggyback, and experimented using a real dataset. Our results showed that tagging systems are quite vulnerable to attack. The results concerning the focused overload attack showed that a goal-oriented attack which targets a specific user group can be easily injected into the system. While producing this attack does not need a lot of effort or knowledge from an attacker's perspective, it may be more difficult to detect this kind of attack since it resembles the natural behavior of a person who is interested in a specific topic. The results of the experiments on different partitions of the data showed that low frequency URLs are vulnerable to piggyback attacks as well as popular and focused overload attacks. These simple attacks should not be ignored and need more attention from system administrators. While high-frequency URLs are robust to piggyback attacks, they are vulnerable to overload attacks.

In our future work, we will model other attack types and compare their impacts on the system. We plan to investigate additional metrics for measuring the impact of an attack, including global measures that more accurately measure the relative prominence of nodes in the tagging network. Understanding the attacks and their effects on social tagging systems will help us discover more robust tagging systems and also guide us to find algorithms for detection and prevention of attacks.

References

1. Burke, R., Mobasher, B., Zabicki, R., Bhaumik, R.: Identifying attack models for secure recommendation. In: *Beyond Personalization: A Workshop on the Next Generation of Recommender Systems*, San Diego, California (January 2005)
2. O'Mahony, M., Hurley, N., Kushmerick, N., Silvestre, G.: Collaborative recommendation: A robustness analysis. *ACM Transactions on Internet Technology* **4**(4) (2004) 344–377
3. Lam, S., Reidl, J.: Shilling recommender systems for fun and profit. In: *Proceedings of the 13th International WWW Conference*, New York (May 2004) 393–402
4. Mobasher, B., Burke, R., Bhaumik, R., Williams, C.: Towards trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Transactions on Internet Technology* **7**(4) (2007) 23
5. Mobasher, B., Burke, R., Bhaumik, R., Sandvig, J.J.: Attacks and remedies in collaborative recommendation. *IEEE Intelligent Systems* **22**(3) (2007) 56–63
6. Williams, B., Mobasher, R.B.: Defending recommender systems: detection of profile injection attacks. *Service Oriented Computing and Applications* **1**(3) (2007) 157–170
7. Xu, Z., Fu, Y., Mao, J., Su, D.: Towards the semantic web: Collaborative tag suggestions. In: *Collaborative Web Tagging Workshop in conjunction with the 15th WWW Conference*, Edinburgh, Scotland (May 2006)
8. Koutrika, G., Effendi, F., Gyöngyi, Z., Heymann, P., Garcia-Molina, H.: Combating spam in tagging systems. *Proceedings of the 3rd international workshop on Adversarial information retrieval on the web* (2007) 57–64
9. Krause, B., Hotho, A., Stumme, G.: The anti-social tagger- detecting spam in social bookmarking systems. In: *Proc. of the Fourth International Workshop on Adversarial Information Retrieval on the Web*. (2008)
10. Heymann, P., Koutrika, G., Garcia-Molina, H.: Fighting spam on social web sites: A survey of approaches and future challenges. *IEEE Internet Computing* **11**(6) (2007) 36–45
11. Sandvig, J., Bhaumik, R., Ramezani, M., Burke, R., Mobasher, B.: A framework for the analysis of attacks against social tagging systems. In: *Proceedings of The 6th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems, Held in conjunction with the 23rd National Conference on Artificial Intelligence - AAAI 2008*, July 13-17, 2008 - Chicago, Illinois, USA. (2008)
12. Schmitz, C., Hotho, A., Jaschke, R., Stumme, G.: Mining association rules in folksonomies. *Data Science and Classification: Proceedings of the 10th IFCS Conference*, Ljubljana, Slovenia, July (2006)
13. Mika, P.: Ontologies are us: A unified model of social networks and semantics. *Web Semantics: Science, Services and Agents on the World Wide Web* **5**(1) (2007) 5–15
14. Riska, A., Diev, V., Smirni, E.: Efficient fitting of long-tailed data sets into hyperexponential distributions. In: *IEEE Globecom Conference, Internet Performance Symposium*, Taipei, Taiwan, November 2002. IEEE Catalog Number: 02CH3798C. (2002)
15. Hotho, A., Jschke, R., Schmitz, C., Stumme, G.: Information retrieval in folksonomies: Search and ranking. In: *Proceedings of the 3rd European Semantic Web Conference*. LNCS, Budva, Montenegro, Springer (June 2006) 411–426
16. Lui, A.: Web information retrieval in collaborative tagging systems. In: *Proceedings of the WI 2006. IEEE/WIC/ACM International Conference on Web Intelligence*. (18-22 Dec 2006) 352 – 355
17. Salton, G., Wong, A., Yang, C.: A vector space model for automatic indexing. *Communications of the ACM* **18**(11) (1975) 613–620