The Copied Item Injection Attack

Nathan Oostendorp
School of Information
University of Michigan
Ann Arbor, MI 48104
oostendo@umich.edu

Rahul Sami
School of Information
University of Michigan
Ann Arbor, MI 48104
rsami@umich.edu

ABSTRACT
In many web communities, users are assigned a reputation based on ratings on their past contributions, and this reputation in turn influences the recommendation level of their future contributions. In this type of system, there is potentially an incentive for authors to copy highly-rated content in order to boost their reputation and influence within the system. We describe this strategy as a copied-item injection attack. We conduct an empirical study of this attack on the online news discussion forum Slashdot. We find evidence of its use and demonstrate its effectiveness in eliciting high ratings. We explore variants of this attack in other domains and discuss potential countermeasures..

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]: Information Filtering

General Terms
Reliability, Security.

Keywords
Manipulation, Recommender System, Online Discussion, User-Contributed Content

1. INTRODUCTION
Numerous online communities offer the ability to post and view user-contributed content, but participants can suffer from information overload in high-traffic environments. Often, rating and filtering systems are used to promote content that has been created or rated highly by leading users in the community. With these systems, an item's initial prominence is often based on the reputation of the content creator. This reputation is based (at least in part) on feedback on items that user has created in the past, and serves as a signal of quality as well as an incentive to improve quality. For example, reviews by 'Top Reviewers' on ePinions [5] and 'Elite Members' on Yelp [22] are prominently displayed, and comments by Slashdot [21] members with higher 'Karma' start at a higher level than other comments.

For these systems, as with other recommender systems, there is increasing concern about manipulation by users with a vested interest in promoting or burying certain target items. There is a growing literature on addressing the threat posed by attackers who create multiple shill or sybil accounts, and then use them to rate items in patterns (perhaps randomized) that will lead to collaborative filtering algorithms boosting or burying the target items. Defense techniques that have been developed include detecting and removing anomalous user profiles [2, 17, 12, 20,13], limiting the influence of user profiles until they have made contributions [19], and providing monetary incentives for honest rating [14, 1]. In this paper, we identify a new class of attacks that user-contributed content recommenders may be vulnerable to: the injection of duplicated or plagiarized items. We study the prevalence and effectiveness of this attack using a corpus of over 20 million comments from the technology news website Slashdot, and propose countermeasures against this attack.

Execution of a copied-item injection attack involves a two-step process for the attacker: First, she must find old items (comments, on Slashdot) that have been rated very highly by the community. She can then duplicate the entire item, or a portion of the item, and post this to the site as a new item (i.e., a new comment on a different story), claiming to be the creator. Site moderators do not always recognize this as a recycled item, and so rate it highly based on the quality of the original item. In turn, this leads to the reputation of the attacker being increased, as she is the purported author of high-quality content. Subsequently, she can exploit this higher reputation, and the improved visibility it brings, to attract attention to subsequent original (and possibly inferior) items she creates.

The first question raised by attacks of this form is: Are they harmful to the site or the rest of the community? This is not obvious, because in some contexts it may be a useful contribution to redirect the community's attention to valuable information that was known in the past, but has been forgotten. For any given domain, this will need to be weighed in comparison to the harm caused by the attack. In section 5, we argue that, for the Slashdot domain, the potential damage caused by this attack outweighs the potential benefit.

Existing techniques to prevent or limit manipulation in recommenders do not protect against copied-item injection attacks. This attack does not require the attacker to change her rating profile, so procedures that detect and filter anomalous ratings would not work. The influence-limiting approach also is not effective: Creating a good duplicate, or rating it highly, will be counted as a contribution by the attacker, but in this context, the attacker is merely reusing earlier information from raters on the original item to infer that the copy will be well-liked, but is not

actually contributing new information. Existing mechanisms that prescribe monetary incentives to rate honestly do not address this problem, as the raters who rate the copies highly are being honest about their perceptions of its quality. In section 6 we discuss some techniques that could be used to combat copied-item injection attacks.

The rest of this paper is structured as follows. In section 2, we review the related literature. In section 3, we formalize our definition of copied-item injection attacks. Section 4 describes our empirical analysis of this attack on the Slashdot dataset, and our measurements of the current prevalence and effectiveness of this attack in the Slashdot domain; we discuss the consequences of these results in Section 5. In section 6, we discuss countermeasures against this threat. We conclude and identify directions for future work in section 7.

2. RELATED WORK

Recently, there has been a rich literature centering on the vulnerability of collaborative filtering recommender systems to attack, as well as defenses against those attacks. This was initially observed by Lam and Riedl [8] and O'Mahony et al. [16]. This literature has focused on a particular class of threats: attackers can create "shill" or "sybil" user profiles, and use these to promote or bury items they have a vested interest in. A number of authors have studied variants of this attack, as well as defenses against them; we refer readers to recent surveys by Mobasher et al [15] and Mehta and Nejdl [13]. Techniques to defend against this attack include methods to detect and remove anomalous user profiles [2,17, 12, 20, 13], limiting the influence of user profiles until they have made contributions [19], and providing monetary incentives for honest rating [14, 1]. The chief difference with our current work is that we consider a different class of attack: we study settings in which, in addition to potentially injecting shill user profiles, the attacker can inject items with known quality (derived by copying existing items).

There has also been prior research on the Slashdot moderation system. Lampe and Resnick [11] analyze the performance of the moderation system in identifying high-quality comments, and show that it is largely effective. Lampe and Johnston [9] report that new users of the site use the moderation feedback they receive as a cue to learn the norms of the community. Lampe et al. [10] propose to use a second level of collaborative filtering to adapt users' interface views of the moderated comments. Poor [18] argues that Slashdot is an archetypical public sphere on the Internet, and describes the role of the Slashdot moderation system in fulfilling this function.

David and Pinch [3] conducted a qualitative study of strategic reviewing on Amazon.com. They document several cases of plagiarized reviews; one of the motives they identify for plagiarizing is to build up a long profile of ratings with low effort. This is similar to the modus operandi of our copied-item injection attack, except that the community's ratings on the content are more important than the raw number of comments in our setting.

3. MODEL AND TERMINOLOGY

In this section, we introduce terminology to clarify our discussion of the copied-item injection attack.

There is a set U of users; we use h to denote an honest contributor, and a to denote the attacker. A set of items I; each item $i \in I$ has two characteristic features: $creator(i)$ denotes the user who is listed as the creator of the item, and $content(i)$ is a description of its content (text, image features, etc.). The attacker has some target content T that she would like to promote. At any point in time, an item has a recommendation level $rec(i)$. For simplicity, we assume that the recommendation level is not personalized; for personalized recommenders, $rec(i)$ could denote the average recommendation level among the target community, or another summary statistic.

Each user u has a reputation $R(u)$. Item recommendation level $rec(i)$ is assumed to depend on its creator's current reputation $R(creator(i))$ as well as the corpus of ratings on the item set I. The user reputation $R(u)$ is assumed to be computed based on the corpus of ratings; we assume that, other things being equal, $R(u)$ is higher if a particular item i with $creator(i)=u$ has higher recommendation level $rec(i)$. We assume that two items i,j with $content(i)=content(j)$ have positively correlated recommendation levels, because the raters cannot consistently identify the later item as having duplicated content. This is realistic in a system with a large number of items and users.

A **copied-item injection attack** involves the attacker copying a genuine item i, with a high $rec(i)$, to create a new item c, with $content(c)=content(i)$, but $creator(c)=a$ while $creator(i)=h$. The attacker then waits for c to collect a sufficient number of ratings, so that $rec(c)$ increases towards the high level of $rec(i)$. Finally, attacker a creates a new item t with $creator(t)=a$ and $content(t)=T$.

The simplest measure of the success of an attack is the difference between $rec(t)$ after this attack then it would have been if item c was not created. A slightly more nuanced measure, which is natural is the context of analyzing a's incentives, is the increase in a's *net benefit*, accounting for the cost of creating the copy c and the opportunity cost of not creating an original posting instead. We explore this idea further in section 6.

4. ANALYSIS OF SLASHDOT

Slashdot is a high traffic online news site and an active forum that receives several thousand user-contributed comments and over a million pageviews every day [20]. To help the users navigate among the large amount of user-contributed material, it uses a rating/moderation system that lets them filter comments based on a score from -1 to 5. This system has elaborate controls to detect and discourage abuse, including rules on who can moderate, how often they can moderate, and how much they influence the score[11].

4.1 Slashdot's Moderation System

The system revolves around two scores assigned to user accounts: **karma**, which is accrued by contributing comments and receiving positive moderations on those comments, and **mod points**, which allows users to rate other users comments up or down. In this system, the users and items are linked by authorship, so that each item's rating is aggregated into karma for the user. A user's karma then determines both the probability of acquiring mod points and the starting score for their posted comments. Because positive ratings on an authors comments gives the author additional influence within the system, there is clear incentive to manipulate the system if a users goal is to gain influence or prominence in these discussions.

Additionally, users can **meta-moderate** and judge whether a users mod points have been spent appropriately. In this process, users can view comment moderation pairs and give a up/down feedback on if each moderation was appropriate. Users who frequently are evaluated as having rated inappropriately become less likely to receive mod points. This was designed to defend against simple manipulations where mod points were traded or spent on inferior comments for the express purpose of improving another users karma.

While this system has some algorithmic checks for basic profile-injection strategies such as detection of high-traffic cyclical moderation patterns between users, there are some manipulation strategies that can be used to gain undue influence within the system. The online comic WellingtonGrey has humorously documented a few of these in flowchart form [6]. This chart identifies tactics for accruing karma including profile-injection ("a second account with mod points"), strategically expressing popular sentiments in comment text ("Is it about Microsoft? Say they suck. Is it about Apple? Say they rule."). It also advises recycling of old material. ("Do you have any old +5 posts on this topic? Quick, post one!") This third tactic describes copying an item to gain positive ratings, and therefore karma.

The Slashdot environment is likely to be an ideal environment for this type of attack, due to several factors. Its longevity as a news source (it celebrated its 10th Anniversary in 2008), and high volume of traffic gives it a large library of existing comments that could be recycled. Since so many comments are posted every day, it is also reasonable to assume readers will be unable to recognize an older comment out of the millions authored on the site. Additionally, the nature of "news cycles" means that certain topics recur frequently: a subject line search shows that Slashdot has over 200 stories on Windows Vista, which has been in the news for 2-3 years.

Based on these factors we can make a few generalizations about where a copied-item attack might be used. Certainly it must have an environment where the cost of item creation is low and also the cost of copying an item is similarly low. The incentive to use the attack must come from when the author receives some indirect benefit from positive ratings on the items they create. The Copied Item attack will also be easier where there are extremely large numbers of items so that the probability of duplication detection by recognition from readers is low. Finally, it will be easier to deploy the attack when items have simple data structures, such as a comment with a block of text, a subject line, and an authorship reference, as opposed to items that might be indexed on many different attributes and therefore may have too many similar attributes to the original.

4.2 Description of Slashdot Data

We used a snapshot of Slashdot's database from January 28, 2009, which contained 20,830,313 comments contributed by 307,158 users across 158,867 news story discussions. Each comment record contained a short subject line, a longer message body, a timestamp of publication, the final rating for the comment, and numerical ids referencing for the story and author.

The rating distribution for comments, shown in Figure 1, is roughly a right-skewed normal distribution centered on the mean of 1.158 with a standard deviation of 1.149. 1.30 million comments have a rating of 4 or 5, or about 6.2% of the entire population.

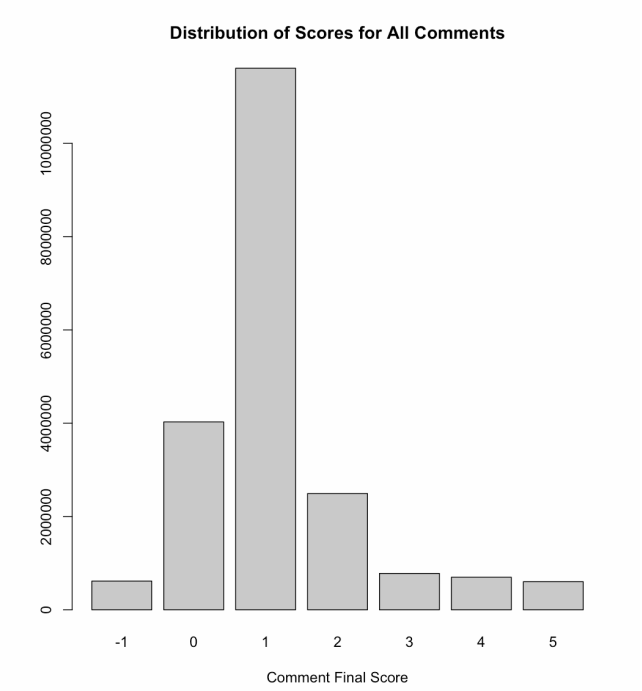

Figure 1: Score Distribution for All Comments on Slashdot

The comment text length distribution is shown in Figure 2 and follows a log-normal distribution. After a logarithmic transformation, the mean comment length is 5.68 (293 characters) with a standard deviation of 1.11. The entire body of text from all of these comments is roughly 11.0 billion characters.

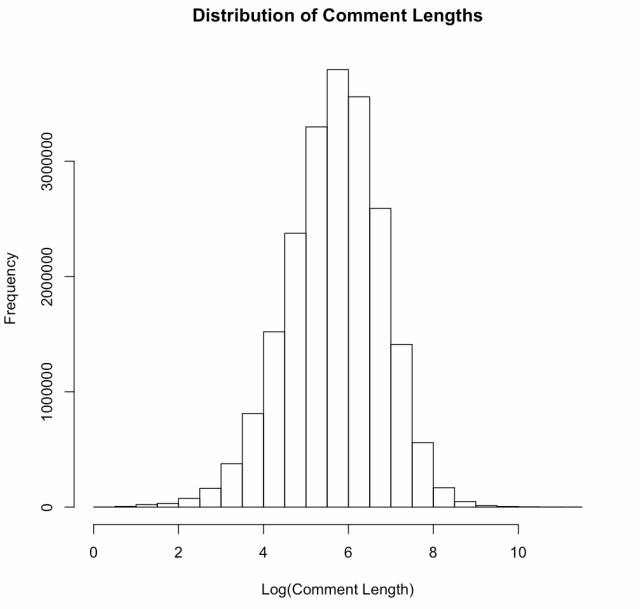

Figure 2: Histogram of Log-Transformed Comment Lengths

4.3 Detection of Copied Items

In this study, our goal was to detect plagiarized comments in this large Slashdot comment corpus. The core of this process was finding comments that shared large substrings. However, there are several conflating factors which could legitimately lead non-attackers to reuse large substrings within their comments: users quote from earlier comments or quote the same source; there is a form of political activism that involves posting the same text repeatedly such as the DeCSS decryption codes; and some users attempt to disrupt a forum by posting as many junk comments as possible. We processed the comments conservatively, so that we would identify a comment as plagiarized only if none of the conflating factors is a plausible explanation for the duplicated text.

In order to detect plagiarisms our first step was to detect comments that had significant duplicate text. We implemented a Rabin-Karp search [7] with a window of 255 characters. Using this method we converted each 255-character substring of a comment message body into a hash value, and searched for co-occurrences of hash values across multiple comments. The entire corpus generated about 6.4 billion (hash,comment_id) pairs. Any comment found to have more than 3 hash collisions with any single previously posted comment was logged. We then went through the logged comment pairs and confirmed that there was significant duplicated text using a longest common substring algorithm. This process resulted in 196,349 pairs of potentially plagiarized comments among the 20 million comment corpus.

In order to narrow this set of comment pairs to distinguish comments that may have been directly plagiarized with intent to boost ratings, we applied a sequence of filtering steps to the original set of copied items. These included:

1. We removed any pairs where the original comment had a final rating score of 3 or less. This was eliminate comment copies that had little reason to expect a high rating.
2. We removed any pairs where the longest common substring was less than 90% of the copied comment length. This was to avoid comments which had significant original material as well as copied content.
3. We eliminated comment pairs where the copied comment did not begin with the longest common substring. This rule was used to weed out quotations since attributions or quotation marks would typically prefix a quote.
4. We removed any comment pairs which appeared in the same story. This was to avoid implicit quoting within replies.
5. We eliminated comment pairs where the copied comment was posted anonymously, rather than by a logged in user, as anonymous users see no direct benefit from having their post rated highly.
6. We eliminated comment pairs where the original comment was copied more than once. , which was used to control for overt reposting, DeCSS code posts, or other forms of habitual reposting.

With these conservative restrictions in place, the set of probable plagiarisms was 735 comment pairs, where 423 users had posted the copied comments. We visually inspected about two dozen pairs manually to confirm that there was no other apparent reason for duplication.

4.4 Hypotheses and Results

Intuitively, we expect that copies of highly-rated comments will also garner high ratings and be useful to potential attackers for the purpose of acquiring karma. In this section we formulate three hypotheses that test this conjecture.

Hypothesis 1: Copying a comment with a high rating is profitable for attackers, in that it produces a comment which is more likely on average to be highly rated.

If the copying of comments were profitable for an attacker, we would expect the copies of these high scoring comments to garner higher ratings than the population at large. We found in the target population of likely plagiarized comments the rating distribution of the copied comments was substantially changed versus the distribution of the global population, as illustrated in Figure 3. Indeed population of copied comments had a mean of 2.15 vs the global mean of 1.16, nearly a full standard deviation higher than the global mean, a difference of 0.987 points. Additionally, 30.4% of items in the copied set had a rating of 4 or 5 as opposed to 6.2% of the global comment population. A two-sample t-test confirmed significance of both results ($p < 0.001$). This discrepancy confirms Hypothesis 1.

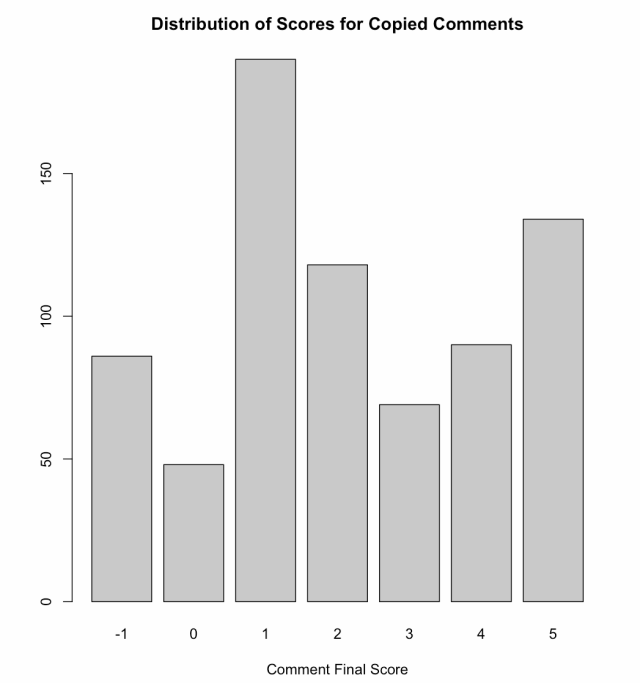

Figure 3: Distribution of Scores for Copied Comments

Hypothesis 2: Copying a comment with a high rating is more profitable than contribution of other content by the attacker.
To see if this strategy is incentive compatible for the attacker, we looked at our set of copied comments compared with the mean rating for the copied item authors other items. By comparing each of the copied comments scores with the users mean post rating in a pair-wise t-test, we found the copied item had a mean improvement of 0.730 points ($p < 0.001$). This confirms Hypothesis 2.

Hypotheses 1 and 2 confirm that copies of highly-rated comments tend to be rated highly even when taken out of their original context. It is conceivable that these comments add value to the

readers of multiple topics, and that little damage is done by rewarding the copiers for reposting them. We will discuss the harm caused by the copied-item attack in more detail in section 5.1. Here, we provide evidence that the copies damage the signaling quality of the Slashdot rating system:

Hypothesis 3: The average rating of comments, other than the copied comment, by the copier is lower than the average rating of other comments by the original poster.

In order to test this hypothesis, we first excluded all instances in which the original comment was posted by an anonymous user. (If the original comment was posted by an anonymous user, we could not identify other comments posted by the same user; further, it is clear to the readers that a comment is anonymous, and hence it is unlikely that they would improve their expectation of other anonymous comments). For each of the 683 surviving instances, we measured the average rating of all comments (other than the copied comment) posted by the original poster, and the average rating of all comments (other than the copied comment) posted by the copier. We find that the average rating for the original poster is 1.70, vs 1.38 for the copiers; a two-sample t-test confirms significance (p <0.001). This suggests that the copiers actually had lower quality than the original posters, and thus, the high rating they receive for the copied content reduces the ability of readers to distinguish them from the higher-quality posters who posted the original comments.

Hypothesis 4: Copied comments are much more likely to be topic starters (comments starting a discussion thread) than other comments, since it would be more difficult to have a copied response seem appropriate as a reply to multiple comments.

We looked at the location of our copied comment population in Slashdot discussions and found that of the 734 copied comments 573 were topic starters. If you contrast this with the entire comment population of 20.8 million, 6.28 million comments started topics. A two-sample t-test indicates that the copied comments are 47.8% (78.0% vs 30.2%) more likely to be topic starting than a comment in general ($p < 0.001$). Hypothesis 3 is therefore confirmed. The consequence of this hypothesis is that copying can distort the pattern of interaction on the site, skewing it towards breadth rather than depth of exchange.

5. DISCUSSION

With H1, H2, and H3 confirmed, it seems evident that item copying has been successfully used on Slashdot to systematically garner high ratings for comments and therefore improve the users karma score. We expect that this type of item injection attack has potential to be a widespread problem both in the realm of Slashdot and other moderation-based comment systems as well as other collaborative filtering spaces. In any forum where inserting copies of highly rated content is incentive compatible and technically possible there is a strong likelihood of abuse. At the core of this incentive problem on Slashdot is the transitive property of item scores to users, where a user stands to directly gain influence in the system by receiving positive feedback on their items. However, systems containing low-cost item creation may present different incentives for this type of attack, and it may create variations in overall impact.

Simple manipulations to try and disrupt this type of behavior may add only marginal costs to the effort required to copy comments. On March 20, 2001 Slashdot deployed a code update that attempted to curtail comment "re-posting" by logging an MD5 hash encoding of the entire comment text. Subsequent comments that were posted with the same MD5 sum as a previous comment were rejected from the discussion. We looked at our copied comments sample set and found 28 comments posted before this feature was deployed, 26 of which were exact copies. After this change it was not possible to post the *identical* comment again; however, it was possible to make a trivial change to a comment, such as addition of whitespace, and repost. Of the 707 copies detected dated after the March 20, 2001, 618 were identical to the original except for the insertion or deletion of punctuation and/or whitespace. After controlling for whitespace and non-alphanumeric characters we found no significant difference between entire/partial match ratio between the two populations using a binomial test.

We suspect that this may possibly be due to the extreme ease with which a duplicated post could be altered by adding even a single whitespace character anywhere in the text. It also may be that our conservative heuristics used to detect likely plagiarisms select primarily towards exact matches in this data set.

5.1 Is Slashdot comment copying really harmful behavior?

From a certain perspective, it may be reasonable to point out that the copied comments on Slashdot do add value to the system. In a sense, the positive ratings that the duplicated comments receive are signals from the raters that the comment has value, and this may add insights that otherwise wouldn't be seen in this discussion environment. While it may take a certain moral flexibility to ignore the taboo of plagiarism, the copied item posters could be thought of as agents of conversational arbitrage, seeking out and shining up old gems from previous discussions. However, simply looking at the reposted comments as harmless injections ignores other externalities of having unattributed reposting in a discussion system. Although the user ratings reflect the immediate visceral reaction of the raters to the content, this may not capture the entire value of a piece of content to the system.

For the Slashdot domain, we believe that the potential damage outweighs the potential benefit: Users can always jog the community's memory by quoting earlier comments with attribution instead of resorting to plagiarizing comments, and quoting is fairly widespread, so there is little additional benefit accrued through these attacks. In fact, given the availability of quoting as an alternative which meets community norms and requires negligible additional effort by the copier, the fact that a user would choose to not credit the original author is illuminating: it indicates that they expect to gain a better reception (and better ratings) by suppressing the fact that the content was duplicated. This in itself suggests that the ratings are not perfectly aligned with the community's perception of the long-term value of a contribution.

It is likely any systemic method to gain karma would have an undesirable effect on the Slashdot system, and become increasingly widespread if the technique was communicated between users. One problem is this tactic distorts karma as a signal of someone who has contributed good fresh content. For instance, in the Slashdot system, karma has a direct impact on the starting score of a users post. Therefore a user with high karma user may start their post at 2, rather than 0 or 1. This means that

the comment ratings lose their effectiveness as a signal of quality as well in this particular situation. This loss of signaling quality was borne out in our confirmation of hypothesis 3.

The other potential impact if this tactic of copying comments was widespread is that it would have a negative impact on the dynamic actual conversations that occur within Slashdot. Hypothesis 4 confirms that these comments tend to be discussion-topic starters, but any replies to these copied comments would be very likely to be disregarded by the attacker. They are, after all talking to a different person than the user who originally generated the comment text. This means in as copied comments became more frequent within the system, the harder it would be for users to find genuinely interactive experiences.

Ultimately, we believe the threat is significant enough that defenses against it merit careful consideration. This phenomenon potentially weakens both incentive and signaling function of the site's reputation system: Users may be incentivized to copy items as a lower-cost way of building reputation than creating original content, even though the latter is a more valuable contribution; and, future original contributions by the attacker may start as a misleadingly high recommendation level, because they reflect the quality of the author of the original item, rather than the attacker's inherent quality. Additionally, it may create an incentive for copying content without attributing the original author, which can disrupt the norms of the online community.

5.2 Variants in other domains

It is possible that a copied item injection attack could potentially appear in other types of recommender spaces where items can be inserted into the system with relatively minor barriers, just as profile injection attacks are potentially problematic in spaces where a user creation in a system is extremely low-cost. In particular, any systems where ratings on items transitively score the users who create the items will provide incentive for this type of attack.

Although the Slashdot recommender system uses a simple voting method of collaborative filtering, it is sophisticated in tracking reputations for users and using these reputations to allocate visibility and influence. Reputation tracking is a powerful method of identifying high-quality contributors over time, so we expect that many recommenders for social web applications will adopt some it in some form. Then, copied-item injection attacks, perhaps in conjunction with other attacks, will become a potential threat.

In particular, it is the combination of an item and profile attack that could be extremely problematic. A sophisticated attacker could use the copied items to establish validity for shill items posted by shill accounts, and likewise rate other comments similarly with shill accounts. This would potentially create a system where scores could be quickly increased on both shill users and items.

In a movie recommender system (or other traditional item recommenders) a combination of an item and user injection could potentially distort recommender predictions if site maintainers were not vigilant about repairing duplication. A copied item, whether legitimately cataloged as a variant of an original film (ie a "directors cut") or sorted under a different name, could be used as a target item in a manipulative attack in order to "push" or "nuke" according to an agents agenda.

Another application in which copying items can increase the power of an attacker is in search engine website rankings. Here, the `ratings' are expressed in the form of other sites linking to a particular site. By copying some content from a high-quality site, an unscrupulous site operator can increase the chances of other genuine sites linking to his site. This will drive up the ranking of his site on search engine results pages; some of these pages can be used to damage readers through unrelated advertisements or fraudulent content.

There are several other domains that could potentially see item-injection attacks. In the news website space, gaming a collaboratively filtered news aggregator such as Digg [3] could be profitable by increasing traffic and therefore ad revenue.

6. POTENTIAL COUNTERMEASURES

In this section, we describe a framework for reasoning about countermeasures to the copied-item injection attack, and identify several possible techniques that could effectively combat this threat. There are two core factors behind the copied-item attack: (1) Users have an incentive to increase their reputation, and incur effort costs when they attempt to do so, either by copying items or by creating fresh contributions. (2) Copied items are likely to garner ratings that are similar to those of the original item. We frame our discussion of countermeasures with these two aspects of the problem in mind.

For a given domain, it is helpful to visualize a space A of possible pieces of content, coupled with a distance metric that captures the similarity between two pieces of content: The smaller the distance between x and y, the more similar the pieces of content. For example, A could be the space of all text strings, and the distance measure could be based on edit distance, or keyword frequencies. For a movie domain, A could be defined by a set of features (title, actors, director, etc.), with a distance metric based on this feature similarity. Modeling the content space in this way allows us to reason about near-copies as well as exact copies. The cost and benefit to an attacker a in executing an item-copy injection attack can then be described in terms of this reference model. When a copies an item i to generate a near-copy item c, her cost is presumably increasing in the distance between $content(i)$ and $content(c)$, reflecting the effort of obfuscating the fact that the item was copied; for example, it takes some effort to reword a comment or change the whitespace and punctuation. The benefit accruing to the attacker depends on the ratings that c garners; given that i was a very highly-rated item, the benefit might be highest for an exact copy but drop off as the distance between $content(i)$ and $content(c)$ increases.

Techniques to combat item-copy injection attacks can work by raising the cost of carrying out the attack, imposing a penalty if the attack is detected, or reducing the benefit of creating the copy item c.

- One natural technique is to detect copies, and either prohibit them outright, or impose a reputation penalty when they are injected. This is the approach that Slashdot implemented when they prohibited exact copies of comments. In practice, however, this imposes an insignificant cost on attackers, as they only have to make trivial changes to a previous comment. Instead of merely identifying exact copies, a slightly more sophisticated approach might detect an item within a certain distance of

a pre-existing piece of content, using a distance metric appropriate for the domain. This has a two-fold advantage: it forces attackers to put in more effort in modifying the original content, and in doing so, the copy is less similar to the original item, leading to a lower expected benefit. Another variation would be to not prohibit near copies, but rather, to merge similar items into a single logical 'item-cluster'.

There are two drawbacks to this approach, however. First, it is only as good as the distance metric used. This might spark an arms race between attackers and site managers, in which attackers continually find clever ways to retain the quality of the original item while appearing to be distant under the current metric, and site managers continuously update the metrics to plug these gaps. Second, as the distance threshold increases, there is a growing threat of false positives: genuine items that get mistaken for copies. This could hamper the contribution of honest users.

- Alternatively, the defense can focus on reducing the benefit to users of copying items, relative to more socially valuable activities such as the creation of original content. The attacker derives benefit because of the increase in her reputation and the priveleges that accompany a better reputation. This suggests that a more sophisticated reputation update may be effective: When a user *a* creates an item *i*, rather than increase her reputation based merely on the average rating of *i*, we should account for the average rating of similar items as well. For example, the creator's contribution might be calculated as the difference between the average rating of item *i* and the average rating of the nearest (in terms of content distance) pre-existing item *j*; or, perhaps, use a similarity-weighted average of all pre-existing items. This reduces the benefit of copying high-quality items, hopefully to the point that users choose more valuable ways of building their reputation. Genuine posting of similar items would still be possible, but there would be a reduced incentive to do so.

  The same approach can be extended to tailor the incentives of *raters* as well as creators. The Influence Limiter [18] scores raters based on the amount they improve predictions for future raters. Loosely, a rater who is the first to rate a high-quality item high will gain the highest score, while subsequent raters will be measured as having diminishing contributions. A rater's accumulated score is then used to limit their influence on others' predictions. In the case of a profile injection attack, the effectiveness of each shill is stunted – as it adds no information, it will not earn a high reputation score, and hence have limited influence. As described in [18], the Influence Limiter might be susceptible to copied-item injection attacks: The attacker expects the copy *c* to have similar ratings to the original *i*, and thus, attacker shills can be the first to put in high ratings where relevant. This can be countered by scoring the early raters on items relative to a benchmark prediction that is the average of pre-existing items with similar content.

- A third technique might be to rely on targeted moderation that flags items as 'legitimate' or 'plagiarized'. Human moderators could be shown nearest content items, and might be more skilled at distinguishing genuine forms of copying from reputation-boosting plagiarism. The tradeoff, of course, is that this requires additional human effort that might be better spent in creating or rating items. In addition, as with rating systems, there would need to be a system to prevent attacker shills from controlling this moderation process.

One constraint on all of these techniques is that calculating distances between pieces of content in a large database can be very computationally intensive. This might preclude the use of these techniques in a online mode. Instead, the automated techniques could be used offline to periodically filter items or adjust reputations. Human moderators trying to locate similar pieces of content online would have to rely on simple distance metrics.

It is not possible to meaningfully evaluate the performance of these techniques on our existing dataset, as the attackers are likely to adapt the detailed form of attack once a specific countermeasure has been deployed. This is borne out by the way in which users sidestepped Slashdot's check for identical comments, as described in section 5. The evaluation of the relative effectiveness of these countermeasures is therefore left as a subject for future work.

7. FUTURE WORK

In this paper, we have identified a class of attacks, copied-item injection attacks, that user-generated content recommenders on the web may be vulnerable to. We have studied this attack in a single domain, but the attack pattern is relevant to many different settings; likewise, countermeasures developed in one setting will be helpful in others as well. There are several important directions for future work. The development and implementation of practical countermeasures should be a priority for applications where the copied item injection attack is a feasible strategy. For some domains where duplicate detection of content is impractical, one direction of research may be to use patterns of user ratings to identify similarity between items.

Additionally, it would be useful to conduct empirical or experimental measurement of the prevalence of this attack in other domains. This would give confirmation as well as a broader understanding of attack patterns and the motivations of attackers.

Once countermeasures have been implemented and deployed, and users have had a chance to adapt to them, it will be important to experimentally determine their effectiveness by comparing the frequency and impact of attacks with and without defenses.

8. ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under award IIS-0812042. We would also like to thank the Slashdot Engineering team at SourceForge Inc, specifically Rob Malda, Jamie McCarthy, and Uriah Welcome for their help in accessing and interpreting Slashdot comment data. We are also grateful to Paul Resnick at the University of Michigan for his helpful feedback and suggestions on this project.

9. REFERENCES

[1] R. Bhattacharjee and A. Goel. Algorithms and incentives for

robust ranking. In Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA '07), 2007.
[2] P.-A. Chirita, W. Nejdl, and C. Zamfir. Preventing shilling attacks in online recommender systems. In WIDM 05, pages 67–74, 2005.
[3] S. David and T. Pinch. Six degrees of reputation: The use and abuse of online review and recommendation systems. First Monday, 6, 2006.
[4] Digg, 2009. http://www.digg.com.
[5] Epinions, 2009. http://www.epinions.com.
[6] W. Grey. The slashdot flowchart, 2007. *http://miscellanea.wellingtongrey.net/2007/04/28/slashdotflowchart/*
[7] R. M. Karp and M. Rabin. Efficient randomized pattern-matching algorithms. IBM J. Res. Dev., 31(2):249–260, 1987.
[8] S. K. Lam and J. Riedl. Shilling recommender systems for fun and profit. In Proceedings of WWW '04., pages 393–402, 2004.
[9] C. Lampe and E. Johnston. Follow the (slash) dot: effects of feedback on new members in an online community. In Proceedings of the 2005 international ACM SIGGROUP conference on supporting group work, 2005.
[10] C. Lampe, E. Johnston, and P. Resnick. Follow the reader: Filtering comments on slashdot. In Proceedings of CHI 07 Conference on Human Factors in Computing Systems, pages 1253–1262, 2007.
[11] C. Lampe and P. Resnick. Slash(dot) and burn: Distributed moderation in a large online conversation space. In Proceedings of ACM CHI 2004 Conference on Human Factors in Computing Systems, 2004.
[12] B. Mehta, T. Hoffman, and P. Fankhauser. Lies and propaganda:detecting spam users in collaborative filtering. In Proceedings of IUI'07, 2007.
[13] B. Mehta and W. Nejdl. Attack resistant collaborative filtering. In Proceedings of ACM SIGIR '08, 2008.
[14] N. Miller, P. Resnick, and R. Zeckhauser. Eliciting honest feedback: The peer-prediction method. Management Science, 51(9):1359–1373, 2005.
[15] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams. Towards trustworthy recommender systems: An analysis of attack models and algorithm robustness. ACM Transactions on Internet Technology, 7(2):1–40, 2007.
[16] M. O'Mahony, N. Hurley, and G. Silvestre. Promoting recommendations: An attack on collaborative filtering. In Proceedings of the 13th International Conference on Database and Expert System Applications, pages 494–503. Springer-Verlag, 2002.
[17] M. P. O'Mahony, N. J. Hurley, and G. C. M. Silvestre. Detecting noise in recommender system databases. In Proceedings of the 2006 International Conference on Intelligent User Interfaces, pages 109–115, 2006.
[18] N. Poor. Mechanisms of an online public sphere: The website slashdot. Journal of Computer-Mediated Communication, 10(2), 2005.
[19] P. Resnick and R. Sami. The influence limiter: Provably manipulation-resistant recommender systems. In Proceedings of the ACM Recommender Systems Conference (RecSys07), 2007.
[20] J. Sandvig, B. Mobasher, and R. Burke. Robustness of collaborative recommendation based on association rule mining. In Proceedings of the 2007 ACM Conference on Recommender Systems, 2007.
[21] Slashdot, 2009. http://www.slashdot.com.
[22] Yelp, 2009. http://www.yelp.com.