Recommender Systems: Attack Types and Strategies

Michael P. O'Mahony and **Neil J. Hurley** and **Guénolé C.M. Silvestre**
University College Dublin
Belfield, Dublin 4
Ireland
michael.p.omahony@ucd.ie

Abstract

In the research to date, the performance of recommender systems has been extensively evaluated across various dimensions. Increasingly, the issue of robustness against malicious attack is receiving attention from the research community. In previous work, we have shown that knowledge of certain domain statistics is sufficient to allow successful attacks to be mounted against recommender systems. In this paper, we examine the extent of domain knowledge that is actually required and find that, even when little such knowledge is known, it remains possible to mount successful attacks.

Introduction

Recommender systems help users to find relevant products from the vast quantities that are often available. In the commercial world, recommender systems have known benefits, e.g. turning web browsers into buyers, the cross-selling of products, instilling customer loyalty, etc. In the research to date, the performance of recommender systems has been extensively evaluated across various dimensions, such as accuracy, coverage, efficiency and scalability. Recently, the security of recommender systems has received consideration from the research community (O'Mahony 2004; Lam & Riedl 2004; Burke *et al.* 2005). It is difficult, given the open manner in which recommender systems operate, to prevent unscrupulous users from inserting bogus data into a system. We refer to the insertion of such data as an *attack*.

Automated collaborative filtering (ACF) algorithms are a key component of recommender systems (Resnick *et al.* 1994; Breese, Heckerman, & Kadie 1998). In previous work (O'Mahony 2004; Lam & Riedl 2004), it has been shown that user–based ACF algorithms are vulnerable to the insertion of biased data. While this work has shown that very effective attacks can be mounted against ACF systems, the attack strategies relied on the attacker having access to certain domain knowledge. We have argued that such knowledge is easy to obtain for many real application scenarios. In this paper, we will examine this aspect of the attacks more closely. In particular, we vary the amount of domain knowledge that is available in order to quantify the amount and type of knowledge required. We also propose a new attack strategy, wherein a recommender system is probed and the system's output is used to discover attack data.

Attack Strategies

Two attack types are considered in this paper– namely *product push* and *product nuke* attacks. The objectives of these attacks is to promote or demote the predictions that are made for targeted items, respectively. All attacks are implemented as follows. The attacker assumes a number of identities within the system being attacked, and creates a user profile for each identity. We refer to such profiles as *attack profiles*. It is in this manner that attack data is inserted into a system – no other access to a system's database is assumed.

There are two issues that attackers need to consider when building attack profiles. The first concerns the selection of items from which the profiles are constructed; the second relates to the ratings that are applied to the selected items. We begin by considering some item selection strategies.

Popular Attack Strategy

In terms of item selection, it is useful to consider the structure of a typical recommender system dataset. In many instances, a number of distinct item clusters may be present in the system. For example, in a music domain, examples of such clusters are House, Country, Classical, Easy Listening etc. Continuing with this scenario, what would be the best set of items that an attacker could select to build attack profiles in order to promote (or demote) a new classical music CD? Assuming that it is intractable for attackers to build attack profiles that consist of all domain items, it makes sense to target primarily those users who have purchased classical CD's in the past, since such users are most likely to make further purchases of this particular kind of music. Thus, existing classical music CD's represent good choices with which to construct attack profiles in order to target this particular subdomain of the system.

While the above strategy limits the set of items to be considered to particular subdomains, some subdomains may encompass a large number of items. In these cases, however, attack costs can be minimised by choosing *popular* items from the subdomain – i.e. items which have received many ratings from users. There are a number of advantages in choosing popular items when building attack profiles:

- the likelihood of a high number of co–rated items between genuine and attack profiles is increased, which helps to ensure high similarities,
- it is desirable that each attack profile have a high probability of being located in the neighbourhood of many genuine users, thereby minimising the cost of attack in terms of number of attack profiles that need to be created, and
- popular items tend to receive consistent and predictably high ratings from users.

The strategy of using popular items applies to all domains, irrespective of the degree of connectivity between domain items. In (Mirza, Keller, & Ramakrishnan 2003), it is noted that users buy and rate different categories of products in qualitatively different ways. For example, movie domains tend to follow a *hits–buffs* distribution, where many users have seen many of the hit movies and where movie buffs, although few in nature, have seen many of the movies. Thus, movie databases tend to be well connected and hence it can be difficult to derive good quality clusters. In other domains, there may be strong connectivities between certain subdomains (e.g. Country and Easy Listening). If such instances are known and regarded by attackers as single item clusters, then the above approach of selecting popular subdomain items for attack profiles remains applicable.

Probe Attack Strategy

Attack profiles constructed using popular subdomain items have the potential drawback of being easy to detect, particularly if large numbers of such profiles are created. While an attacker could attempt to vary the items that are used, nevertheless a distinctive attack signature may exist. Thus, a less conspicuous strategy is desirable. One such approach involves *probing* the recommender system and using the system's recommendations as a means to select items. By rating a small number of initial seed items, the attacker can progressively build up attack profiles which will closely match the distribution of items that have been rated by genuine users of the system. Thus, the probability of high similarities between genuine and attack profiles is assured. Further, any attack profiles that are created are unlikely to be readily distinguishable from genuine profiles, thereby reducing the detectability of the attack.

This strategy also has another advantage over the popular attack, since less domain knowledge is required by an attacker. Only a small number of seed items need to be selected by the attacker[1], thereafter the recommender system itself is used to identify additional items.

Ratings Strategy

The ratings strategy that we adopt is tailored to the particular ACF algorithm that is used. Here, we consider the well established user–based algorithm described in (Resnick *et al.* 1994). This algorithm employs a deviation from mean approach to the calculation of predictions, taking into account any differences in rating patterns across different users. A prediction $p_{a,j}$ is computed for a user a on an item j as a weighted average of n neighbours' ratings as follows:

$$p_{a,j} = \bar{r}_a + \frac{\sum_{i=1}^{n} w(a,i)(r_{i,j} - \bar{r}_i)}{\sum_{i=1}^{n} |w(a,i)|} \qquad (1)$$

where $\bar{r}_a$ is the average rating of user a and $r_{i,j}$ is the rating assigned by neighbour i to item j. The similarity, $w(a,i)$, between the active user a and neighbour i is typically computed using Pearson correlation (Resnick *et al.* 1994).

From (1), the dependence of a prediction on a database d may be expressed as:

$$p_{a,j}(d) = \bar{r}_a(a) + \sigma_{a,j}(d) \qquad (2)$$

Since the term $\bar{r}_a(a)$ depends entirely on the active user, the insertion of attack profiles into a system can only affect the deviation from user mean term, $\sigma_{a,j}(d)$. Therefore, for successful product push and nuke attacks, it is necessary to ensure that the magnitude of the deviation term is maximised or minimised, respectively. The contribution to this term by any particular neighbour is a function of:

- the rating of the item for which a prediction is sought,
- the neighbour's mean rating, and
- the similarity between the neighbour and the active user.

Thus, careful selection of the ratings that are assigned to attack profile items is required. Firstly, attack profiles need to have a high degree of similarity with genuine users if they are to influence predictions. Secondly, since Pearson correlation results in values between -1 and 1, it is important that all attack profiles correlate either positively or negatively with genuine users. Otherwise, the contributions of multiple attack profiles may be canceled out or attacked items may be inadvertently pushed instead of being nuked, or *vice versa*.

Let us consider these issues in the context of the popular attack strategy. By choosing attack profile items that are generally liked and disliked by the genuine users of a particular subdomain, and by rating these items appropriately, the attacker can achieve the above criteria. Consider a push attack, for example, where the objective is to maximise the term $\sigma_{a,j}(d)$ in (2). Our strategy involves assigning the minimum rating, r_{min}, to the disliked items, ratings of $r_{min}+1$ to the liked items and the maximum rating, r_{max}, to the item being targeted by the attack. Thus, positive correlations are achieved between attack and genuine profiles. In addition, the difference term $(r_{i,j} - \bar{r}_i)$ in (1) is maximised for the attack profiles, which is the desired outcome for a successful push attack. In a similar manner, product nuke attacks can be implemented by switching the ratings that are assigned above to the liked and disliked items. This results in negative correlations between attack and genuine profiles and leads to minimising the term $\sigma_{a,j}(d)$ in (2).

For the probe attack, it is only necessary to apply the above strategy to the initial seed items that are selected – thereafter the recommender system is used to select any further items and corresponding ratings.

The difficulty with the above approach lies in identifying popular items which are both liked and disliked by genuine users. In some domains, such data is readily accessible. For

[1]In a sense, seed items need not necessarily be bogus since they may in fact reflect the "true" choices of the attacker.

example, the MovieLens system[2] provides the average rating received by each item in the system. It is also possible to estimate this data from other sources, i.e. by counting the number of positive and negative item reviews, etc. In the results section, we evaluate attack performance when such knowledge is both fully and only partially known.

Experimental Evaluation

Datasets

We use the following real–world datasets to evaluate the attack strategies as described in the previous section.

The EachMovie recommender system[3] operated between 1995 and 1997. The original dataset has some $72,916$ users who provided $2,811,983$ ratings on $1,628$ movies. From this, a random sample of $1,000$ users is selected, which contains $63,507$ transactions on a rating scale of 1 to 6.

The second dataset that is used is provided by the MovieLens research project. MovieLens is a web–based movie recommender system that began operating in 1997. It consists of 943 users, $1,682$ movies and contains $100,000$ transactions in total. Ratings are based on a scale of 1 to 5.

Upon examination, we found that the above datasets were well connected. Thus, we treat each as a single cluster and apply our attack strategies accordingly. In addition, we applied our attack strategies to a dataset obtained from the Smart Radio system (Hayes *et al.* 2002), which is a music recommendation service operated by Trinity College Dublin, Ireland. Due to limitations of space, we omit the results that were obtained using this latter dataset, noting that similar trends were observed using all three datasets.

ACF Algorithm

We use a tuned version of the user–based ACF algorithm that was described above. We employ the k–nearest neighbour neighbourhood scheme, with optimal neighbourhood size chosen by experiment in the usual manner (k is set to 35 for MovieLens and 45 for EachMovie). In addition, we incorporate the significance weighting algorithm extension proposed in (Herlocker *et al.* 1999), which considers the number of co–rated items between users when computing similarity weights. Weights that are based on low numbers of co–rated items are devalued. In this paper, we consider the Pearson correlation similarity metric only. In earlier work (O'Mahony, Hurley, & Silvestre 2003), the performance of various other neighbourhood formation schemes and similarity metrics when subjected to attack was examined. None were found to provide robustness against attack.

Metrics

We introduce the following metrics to evaluate our attacks. For product push attacks, we calculate the increase in the number of *good predictions* (GP) that are made for items following an attack. We define a good prediction for an item j as the number of times the following expression holds true:

$$p_{u,j} : p_{u,j} \geq \delta_g, \forall\, u \in U_j \qquad (3)$$

[2]http://movielens.umn.edu/.

[3]http://www.research.digital.com/.

where δ_g is a threshold rating and U_j is the set of all genuine users who have rated item j.

Likewise, we evaluate product nuke attacks in terms of the number of *bad predictions* (BP) that are made for items after an attack. In a similar manner to the above, we define a bad prediction for an item j using a threshold δ_b as:

$$p_{u,j} : p_{u,j} \leq \delta_b, \forall\, u \in U_j \qquad (4)$$

This metrics are useful measures of attack success since users are likely to act on good and bad predictions in a particular manner – i.e. purchase/do not purchase an item (or at least consider the purchase or non–purchase of an item). In this paper, we set $\delta_g = r_{max} - 1$ and $\delta_b = \frac{1}{2}(r_{max} - r_{min})$. While definitions of what constitutes good and bad predictions are a subjective matter, we believe that those presented here are reasonable and suitable for general analysis.

Experimental Procedure

The following procedure, unless stated otherwise, is adopted in all cases. The performance of an item that is subjected to attack is evaluated over all the genuine users of a system who have rated the item in question. For each of these users, an *all but one* protocol is adopted wherein the test set is comprised of the user–item pair in question, and a prediction for this pair is made using all remaining data. The average result over all users is then computed.

In the results that are shown in this paper, the average performance that is achieved across all the items that are contained in a system is presented. Average performance is calculated by repeating the above procedure for all items, and taking the mean of the results obtained.

Results

Popular Attack Strategy

To begin, we assume that full knowledge concerning both *item popularity* and *item likeability* is available to attackers. Item popularity refers to the number of ratings received by each item and item likeability refers to the average rating received by each item. This scenario in optimal the sense that an attacker can choose the most popular liked and disliked items with which to build attack profiles. Table 1 shows the results for a product push attack on the MovieLens (ML) and EachMovie (EM) datasets. In each case, attack profiles of size 100 items are used. The attacks on both datasets were very successful. For example, the number of good predictions increased to 35% from a pre–attack baseline of 19% when just 1 attack profile was inserted, a percentage increase of 84% (MovieLens). For both datasets, it is apparent that little additional gain was realised by creating more than 16 attack profiles, where the number of good predictions that was achieved was approximately 86%. The results for product nuke attacks are also shown in Table 1. The attacks were, again, very successful. As before, an attack strength of 16 profiles was sufficient to achieve very large effects, with 83% of all predictions being reduced to less than or equal to the midpoints of the corresponding rating scales.

We now examine the effect of attack profile size on attack success. The results, shown in Figure 1, correspond to

Table 1: The effect of the number of attack profiles inserted (#) on the outcome of product push and nuke attacks. The data shown is the number of good and bad predictions that are achieved by the attacks.

	Popular Attack				**Probe Attack**			
	Push GP (%)		**Nuke BP (%)**		**Push GP (%)**		**Nuke BP (%)**	
#	**ML**	**EM**	**ML**	**EM**	**ML**	**EM**	**ML**	**EM**
0	19	24	36	18	19	24	36	18
1	35	46	51	38	26	36	42	31
2	45	55	57	48	32	44	47	37
4	59	67	65	61	40	52	53	45
8	76	79	76	73	49	61	58	55
16	86	87	83	84	59	68	64	63
32	87	89	83	86	66	74	69	70
64	87	89	83	86	71	78	72	74
128	87	89	83	86	74	79	75	77

a constant number (32) of attack profiles inserted into each of the datasets. For both datasets, the increase in the number of good predictions grew with attack profile size. The optimum attack profile size appears to be 100 items, where approximately 88% of all predictions equalled or exceeded the second–highest rating on the respective scales. Thereafter, at greater attack profile sizes, only small increases in the number of good predictions were achieved. Similar trends were found to apply in the case of product nuke attacks.

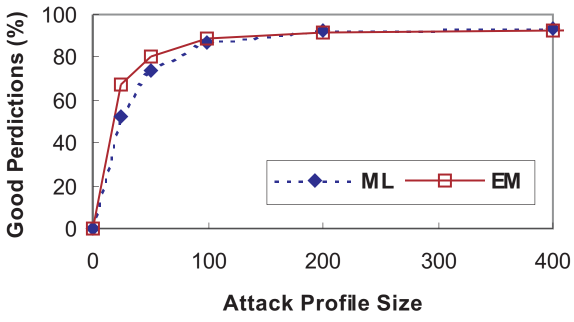

Figure 1: The effect of attack profile size on the number of good predictions that are achieved by a popular push attack.

In the above attacks, it was assumed that full item popularity and item likeability knowledge was known. In the next sections, the effectiveness of the attacks when such knowledge is only partially known is examined.

Item Popularity We begin by assuming that full item likeability knowledge is known and that item popularity knowledge is only partially known. In order to simulate partial popularity knowledge, a number of attacks are implemented where it is assumed that only a certain percentage of the most popular items that are contained in a system are known. In each of the attacks, attack profile items are drawn from the known set of the most popular items. A total of 32 attack profiles of size 100 items are inserted in each attack.

Figure 2 shows the effect of item popularity knowledge on the number of good predictions that are achieved by a product push attack. The percentage of the most popular items from which attack profile items are drawn is shown on the x–axis. In the case of EachMovie, for example, when the top–10% of the most popular items are known, attack profiles are constructed using items which are drawn from the 140 most popular items that are contained in the system[4]. Drawing from the top–100% of the most popular items simulates a complete lack of item popularity knowledge.

As can be seen from the figure, attack success diminished as the lack of item popularity knowledge grew. This result was anticipated, since the overlap between genuine and attack profiles can be expected to reduce as item popularity knowledge decreases. In all cases, however, the post–attack numbers of good predictions were well in excess of the pre–attack levels for both datasets. In addition, even when no item popularity knowledge was assumed to be known, attack success remained high. Consider the EachMovie dataset, for example, where the number of good predictions only fell to 74%, compared to 90% when the top–10% of the most popular items were known. Corresponding values of 64% and 90% were obtained for the MovieLens dataset. These findings are significant, since they indicate that item popularity knowledge is not a necessary requirement for successful attacks. Similar trends were observed for product nuke attacks.

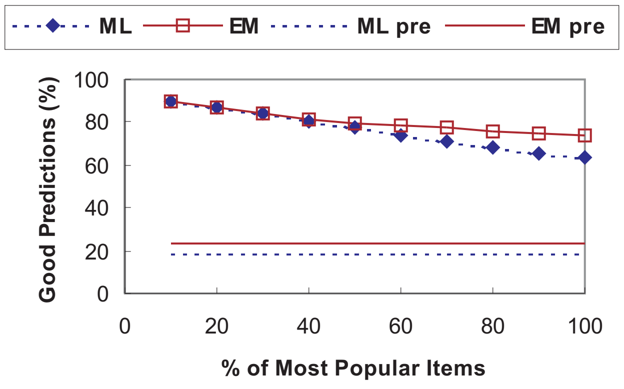

Figure 2: The effect of item popularity on the number of good predictions achieved by a popular push attack. Attack profiles contain items which are drawn from the top–$X\%$ (shown on the x–axis) of the most popular items.

Item Likeability In this section, we investigate the effect of item likeability knowledge on the outcome of a push attack. This knowledge allows an attacker to identify items that are generally liked or disliked in a system. By assigning higher ratings to the liked items, the desired positive correlations between attack and genuine users are achieved.

Here we assume that full item popularity knowledge is known to the attacker. To simulate partial item likeability knowledge, a number of attacks are implemented where a certain percentage of liked and disliked items are misclassi-

[4]From a total of $1,338$ items contained in the sample EachMovie dataset which received at least 1 rating.

fied. For example, a misclassification of 10% means that a randomly selected 10% of the liked items that are contained in attack profiles receive a lower rating and a corresponding percentage of the disliked items receive a higher rating.

The results are shown in Figure 3 for attack profiles containing 100 items. It can be seen that attacks became less successful as the percentage of misclassified items approached 50%. The reason for this trend is that lower similarities between genuine and attack users were achieved at higher misclassification rates and thus, fewer attack profiles were present in neighbourhoods. Nevertheless, it is clear that the attack remained successful for misclassification rates as high as 40%. At this point, 51% of all predictions either exceeded or equalled the second–best rating, compared to 87% when all items were correctly classified (MovieLens). These values are well above the pre–attack baseline number of good predictions, which was 19%.

When 50% of items were misclassified (equivalent to randomly assigning ratings to the liked and disliked items), the effect of the attack was negligable. The reason for this result is that attack profiles were equally likely to correlate positively as they were negatively with genuine users, and thus, no net effect was observed. Misclassification rates in excess of 50% resulted in items being *nuked*, since negative correlations between genuine and attack profiles predominately occurred, which lead to low predictions being made.

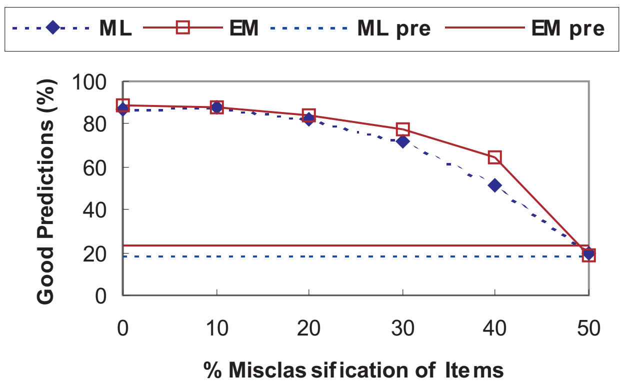

Figure 3: The effect of item likeability on the number of good predictions achieved by a popular push attack.

In summary, the results presented above are significant for two reasons. Firstly, *low–cost* attacks, involving relatively few, small–sized attack profiles, are capable of achieving considerable success. Secondly, the attacks remain successful even when relatively little domain knowledge is known.

Probe Attack Strategy

The key issue in the implementation of a probe attack lies in the selection of the initial seed items. For simulation purposes, we randomly select 10 seed items from the set of the 100 most popular items. It is assumed that item likeability knowledge is known for the seed items. Additional items and corresponding ratings are then selected incrementally by choosing each successive item randomly from the top–N recommended list, which is obtained from the system under attack. In cases where multiple attack profiles are inserted into a system, each attack profile is created independently – i.e. there is no feedback between successive attack profiles. Each attack profile consists of a total of 100 items.

The results for product push and nuke attacks are shown in Table 1. The attacks achieved significant success against both datasets. As before, attack success improved when greater numbers of attack profiles were inserted into the system databases. For example, when 32 attack profiles were inserted, the push attack against the MovieLens dataset resulted in 66% of all predictions equalling or exceeding the second–highest rating, compared to 40% when 4 attack profiles were used. Nevertheless, both of these values were considerably greater than the pre–attack number of good predictions, which was equal to 19%. It can also be seen from the results that little additional attack gains were realised by inserting more than 64 profiles into either dataset.

Figure 4 shows the effect of attack profile size on the outcome of the push attack (similar trends applied for the nuke attack). The results correspond to an attack strength of 64 profiles inserted. As with the popular attack, attack success increased as attack profiles were constructed using more items. In this case, however, attack profiles sizes of up to 400 were required in order to achieve the greatest effects, compared to profile sizes of only 100 for the popular attack strategy. Thus, while the probe attack strategy was successful in substantially influencing the predictions that were made, the popular attack achieved better results in cases where full domain knowledge was known. It is clear, therefore, that the ready availability of such knowledge represents an increased security threat to user–based ACF algorithms.

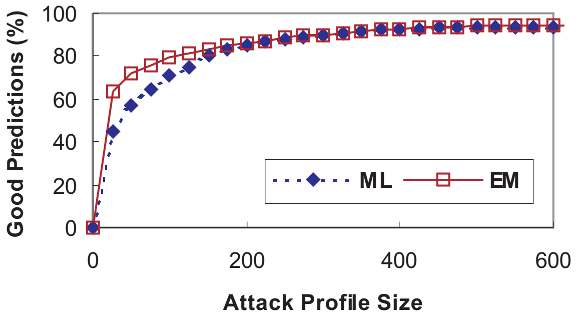

Figure 4: The effect of attack profile size on the number of good predictions that are achieved by a probe push attack.

We now examine the effect of the number of initial seed items that are chosen on the outcome of the probe attack strategy. The results are shown in Figure 5 for both datasets. Experiments were conducted using three different seed profile sizes, ranging from 5 to 15 items. Attack profiles were then incrementally increased in size up to a total of 100 items. In each of the attacks, 64 attack profiles were inserted. As can be seen, there is very little difference in the results between seed profile sizes of 10 and 15 items. Attacks based on 5 seed items performed only marginally worse, with average differences of only 3% seen in terms of the numbers of good and bad predictions that were achieved.

Thus, we conclude that the findings of this section have serious consequences for system security, since domain knowledge concerning only a small number of items is required in order to implement successful attacks.

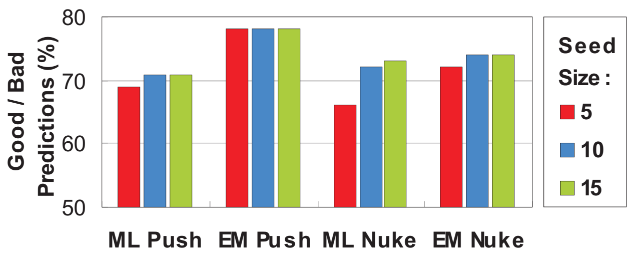

Figure 5: The effect of initial seed attack profile size on the outcome of probe product push and nuke attacks.

Large–Scale Systems

It is important to understand how system robustness varies with dataset size – for example, as the number of genuine users in a system grows, attacks may well become more difficult to implement successfully. Thus, we conducted experiments using different–sized samples drawn from the EachMovie dataset, where identical popular push attacks were carried out on each. Only 1 attack profile, containing 25 items, was used in each case. Since it is reasonable to expect that newer items are likely to be the subject of most attacks (given that such items often sell at premium rates and that more established opinions are held on older, more frequently rated items), the results presented in Table 2 therefore relate to those items which have received ≤ 100 ratings in the various sample datasets. Note that, even at the largest sample size (10,000 users), 52% of all items satisfied this criterion.

The results indicate that the optimum neighbourhood size (w.r.t. predictive accuracy) increased as the sample size grew. This finding was expected since greater numbers of high quality neighbours were likely to be present in the larger sample datasets. Of critical importance is the fact that the attacks remained successful, irrespective of dataset size, where the percentage increase in the number of good predictions that was achieved remained (approximately) consistent, at 46%. We can therefore conclude that dataset size does not have a significant effect on attack success.

Table 2: Popular push attacks carried out on various samples drawn from the EachMovie dataset.

Dataset Sample	# Users	Neigh. Size	% Items with Pop. ≤ 100	% Inc. GP
d_1	2,000	40	79	46
d_2	4,000	50	67	45
d_3	6,000	65	60	47
d_4	8,000	80	55	60
d_5	10,000	95	52	47

Conclusions

In this paper, we have introduced item and ratings selection strategies for attacks against user–based ACF algorithms. Our popular attack strategy achieved the greatest effects in situations where full domain knowledge concerning item ratings was known. We also found, however, that even when little domain knowledge was available, very successful attacks were still possible. The probe attacks also resulted in considerable success, and have the advantages of requiring less domain knowledge to implement and, importantly, attack profiles that are created using this approach are likely to be difficult to detect.

The security implications of the above findings are clear. In future work, we will expand upon the solutions to defend against attack as proposed in (O'Mahony 2004). These techniques aim to exclude from neighbourhoods any biased data that is present in a system. In particular, we introduced the *profile utility* concept, which defines novel approaches to neighbourhood formation and similarity weight transformation. This technique significantly increases the cost (i.e. the number and size of attack profiles required) of successful popular attacks. It needs to be investigated whether this technique will be as successful against other forms of attack. In addition, an investigation of the robustness of various other recommender system algorithms, e.g. content–based, demographic and hybrid approaches, needs to be conducted.

References

Breese, J. S.; Heckerman, D.; and Kadie, C. 1998. Empirical analysis of predictive algorithms for collaborative filtering. *In Proc. of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence* 43–52.

Burke, R.; Mobasher, B.; Zabicki, R.; and Bhaumik, R. 2005. Identifying attack models for secure recommendation. *In Proc. of the Beyond Personalization 2005 Workshop, Int. Conference on Intelligent User Interfaces* 19–25.

Hayes, C.; Cunningham, P.; Clerkin, P.; and Grimaldi, M. 2002. Programme driven music radio. *In Proc. of the 15th European Conference on Artificial Intelligence* 633–637.

Herlocker, J.; Konstan, J.; Borchers, A.; and Riedl, J. 1999. An algorithmic framework for performing collaborative filtering. *In Proc. of the 22nd Int. Conference on Research and Development in Information Retrieval* 230–237.

Lam, S. K., and Riedl, J. 2004. Shilling recommender systems for fun and profit. *In Proc. of the 13th Int. World Wide Web Conference* 393–402.

Mirza, B. J.; Keller, B. J.; and Ramakrishnan, N. 2003. Studying recommendation algorithms by graph analysis. *Journal of Intelligent Information Systems* 20(2):131–160.

O'Mahony, M. P.; Hurley, N. J.; and Silvestre, G. C. M. 2003. An evaluation of the performance of collaborative filtering. *In Proc. of the 14th Irish Int. Conference on Artificial Intelligence and Cognitive Science* 164–168.

O'Mahony, M. P. 2004. Towards robust and efficient automated collaborative filtering. *Ph.D. Dissertation.*

Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom, P.; and J.Riedl. 1994. GroupLens: An open architecture for collaborative filtering of netnews. *In Proc. of the ACM Conference on Computer Supported Cooperative Work* 175–186.