This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2008 proceedings.

A Trust and Reputation based Anti-SPIM Method

Jun Bi, Jianping Wu, and Wenmao Zhang
Network Research Center, Tsinghua University
Beijing 100084, China
junbi@tsinghua.edu.cn

***Abstract*—Instant Messaging (IM) service is a killer application in the Internet. Due to the problem of IM spam (SPIM), building an effective anti-spim method is an important research topic. At present, most of anti-spim solutions are based on email-spam prevention techniques, which are not directly applicable to anti-spim. In this paper, we present a new anti-spim method *SpimRank*, which integrates trust and reputation mechanisms with black-list technique. SpimRank also tracks user's historical action to deal with spim attacks in nearly real time, which is applicable to IM environment.**

***Keywords*- Instant Messaging, SPIM, Reputation**

I. INTRODUCTION

Instant Messaging (IM), as a killer application in the Internet, provides nearly real-time messages delivery and presence information services. However, the problem of IM spam (also known as SPIM) has become a challenge since 2002. [1] indicates that 5%~8% of all enterprise IM today is spim, and IM users are estimated to get an average of 25 spim messages per day by 2008. Due to IM's real time characteristic, spim may cause more severe damages than email spam. Since spim and email spam have some common characters, some anti-spim solutions are based on traditional techniques used to deal with email spam. However, they are not quite suitable for spim because of the differences in system infrastructures and characteristics between IM and email service [2].

The paper is organized as follows: Section II discusses related work. In Section III, we present an anti-spim method *SpimRank*, which is evaluated in Section IV. Section V concludes the paper.

II. RELATED WORK

Currently, spim prevention study is still in an early stage, such as black/white-list that using privacy and contact-lists provided by IM service. Although the black/white-list methods are simple and effective, they are not the ultimate solutions for business environment, because it's hard to deal with new customers who are not on their white or black lists. [2] provides an anti-spim method by integrating a number of mature spam defending techniques with modification and using their new techniques based on limiting IM sending rate. Such method could be deployed on both client and server/gateway to deal with spim attack. In [2], authors evaluate every mature spam defending technique that with modification for IM service. Although spam and spim have some common characteristics, there is significant difference between them, especially the real time characteristic in IM service [2].

Trust and reputation mechanisms are popular in ranking a set of elements, such as web pages or people. Among the algorithms proposed for reputation scheme, the PageRank[3] algorithm is to compute Web-based reputation value. PageRank computes rank values by means of link analysis, i.e., based on the graph inferred from the link structure of the Web. The main idea of PageRank is that "a page has a high rank if the sum of the ranks of its back-links is high". Give a page p, the set of its input links I (p) and output links O (p), the PageRank score is computed according to the formula:

$$PR(p) = c \cdot \sum_{q \in I(p)} \frac{PR(q)}{\|O(q)\|} + (1-c) \cdot E(p) \quad (1)$$

The damping factor $c < 1$ (usually 0.85) is necessary to guarantee convergence and to limit the effect of rank sinks. The $\boldsymbol{E}$ vector has all entries equal to $1/N$, where N is the number of pages in the Web graph. Formula (1) can be also transformed into the form as a matrix computation: where matrix $\boldsymbol{T}$ is a $N*N$ matrix that $t_{ij} = 1/|I(\mathrm{i})|$ if node j has a link to node i, otherwise 0.

$$T' = c \cdot T + (1-c) \cdot E \quad (2)$$

In theory, the vector that represents the rank values is to compute the dominant eigenvector of matrix $\boldsymbol{T}$. In practice, we always use power iteration algorithm to compute an approximate value. The main advantage of these algorithms such as PageRank is that they are designed for high attack resilience.

III. SPIMRANK

Compared with email service, the influence of spim attack is more serious to IM receivers than those encountering with email spam, because of the real time characteristic of IM service. Therefore, it's necessary to detect and filter spim rapidly. In order to achieve such requirement, the design of filtering method should be effective and simple enough.

The black/white-list techniques are simple and effective enough to filter spam (or spim), but may not be appropriate when dealing with strangers that neither in black nor white lists (we call them gray-list users). SpimRank will make use of trust

978-1-4244-2026-1/08/$25.00

and reputation mechanisms to deal with gray-list users when filtering incoming IM messages.

A. Design for Trust and Reputation Mechanisms

The design for trust and reputation mechanisms includes the following steps: (1) Build the reputation network; (2) Design trust and reputation algorithm, and (3) Trust and reputation management.

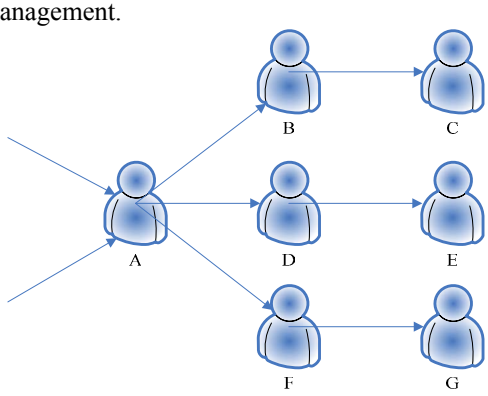

Figure 1. **Gwhit** based on contact-list

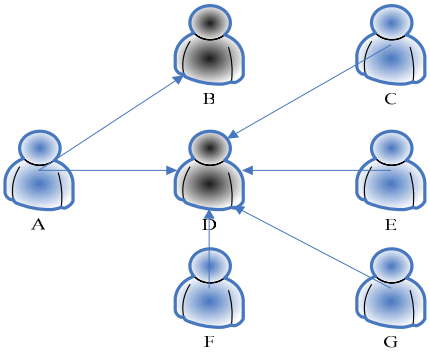

Figure 2. $\mathbf{G_{black}}$ based on black-list

(1) Build the reputation network.

In IM service, contact list information are always stored at server side, which is convenient for server to collect them in order to form a social network. In SpimRank, we collect contact list information from IM server and build our reputation network, as shown as Figure 1. It is a directed graph (we call it $\mathbf{G_{white}}$) where every node represents IM user, and the directed edge, i.e., A→B means user A has added B into his contact list. Such network represents the good votes between users.

Furthermore, we can also form a reputation network based on blacklist filtering techniques, which is supported by most of current IM service. Figure 2 shows the blacklist-based reputation network ($\mathbf{G_{black}}$): the directed edge between node A and D means user A has added D into his blacklist. And such a network represents the bad (or punitive) votes between users.

From directed graph G_{white}, we define an $N*N$ matrix $\mathbf{T_{white}}$, where N is the number of users in G_{white}, and T_{ij} = *1/NumOfVote(i)* if user j has added i into his contact-list (*NumOfVote(i)* is the total count of users who have added i into their contact-list); otherwise 0. We also define a vector $\mathbf{V_{black}}$, where $V_{black}[i]$ equals the amount of users who have added i into their black-lists (we call it **Blacklist Level**).

(2) Trust and reputation algorithm.

We calculate two kinds of reputation values: local trust value and global reputation value. When stranger *C* send IM to *A*, *C*'s local trust value, if exists, will be computed firstly to *A*. Otherwise, a global trust vale will be computed.

Algorithm to global reputation value. The basic idea of algorithm to compute global reputation value in SpimRank is based on PageRank: a user has a high rank if the sum of the ranks of users who add him into their contact-list is high. We substitute $\mathbf{T_{white}}$ for the variable $\boldsymbol{T}$ in formula (2) described in section II. We use power iteration algorithm to calculate the vector that represents the global reputation value.

Algorithm to local trust value. The calculation of the local trust value in SpimRank is based on such principles:

- Trust transitivity [4]. If *A* trusts *B* (means *B* exists in *A*'s contact-list) and *B* trusts *C*, then we can confirm that A will also trust *C* on the recommendation from *B*.
- Compared with the user who has lower reputation value, Users with higher reputation value will influence more heavily when rating to other user.

The algorithm to compute C's local trust value (relative to *A*):

- Find all achievable paths from *A* to *C* within three hops (such as A->B->C or A->B->D->C, but invalid when A->B->D->E->C), named as S_{path}.
- For each *path* in S_{path}, calculating $E_w = (\sum w_i \times T_i)/|path|$, where i is the node next to *A* to the one before *C* in this path, |path| is the amount of node (exclude node *A*) in this path. T_i is the global reputation value to node *i*, and weighted value $w_i = 2^{-(hop(i, A)-1)}$, where hop(i, A) is the number of hop from node *i* to *A*.

For each E_w, calculating $LT_{ac} = [\sum w'_i \times E_w(i)]/|S_{path}|$, where w'_i is the weighted value for $E_w(i)$, and the higher $E_w(i)$, the larger w'_i is. $|S_{path}|$ is the amount of paths that S_{path} contains Finally, the LT_{ac} is the local trust value to *C* (relative to *A*).

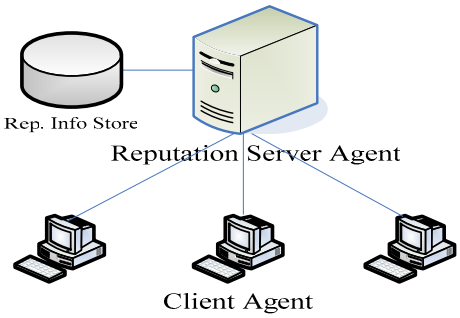

Figure 3. The reputation system deployed in autonomous IM domain.

(3) Trust and reputation management.

Most current public IM services, such as MSN, Yahoo!, do not cooperate with each other. In such autonomous IM service domains, the architecture of *SpimRank* reputation system can be deployed like Figure 3: A **Reputation Server Agent** collects reputation information such as contact-list and black-list from IM server, then compute, store, and distribute the reputation value.

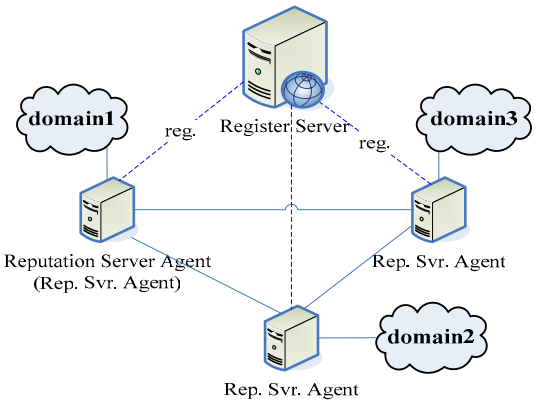

Figure 4. The reputation system deployed in multi-domains.

On another hand, there are also many IM servers based on open IM protocols, such as XMPP protocol. These servers are able to cooperate with each other. In such multi-domain scenario, the architecture of our reputation system can be deployed like Figure 4: a **Register Server** is built to manage registration and authentication of reputation server agents. The register server also collects contact-list and black-list information from those registered agents in order to form a global view of reputation network.

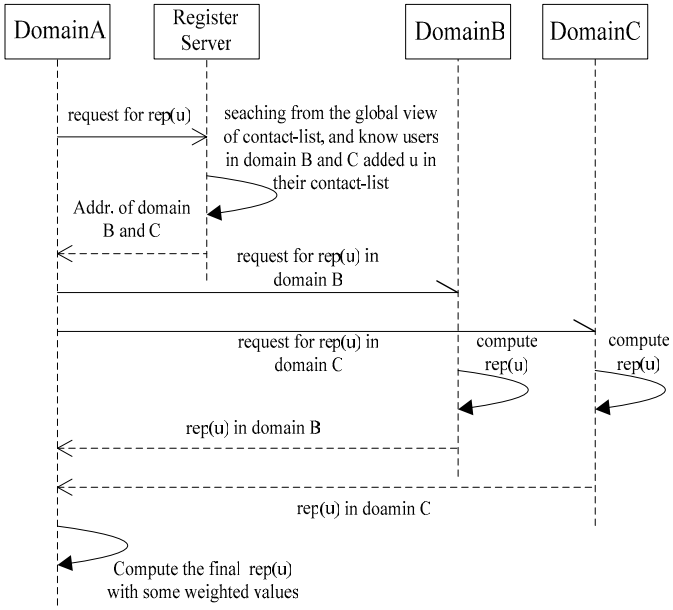

Figure. 5. The diagram of computing reputation value in multi-domains.

When a user's reputation value is needed, the agents and Register Server will communicate with each other in the order shown in Figure 5.

B. Design of Spim Filtering Method

The SpimRank filtering method is described in Figure 6. The incoming IM messages will be firstly filtered by black-list and white-list mechanisms. When dealing with gray-list users, a reputation-based filtering method will be used. In *SpimRank*, the reputation-based filtering method integrates trust and reputation mechanisms with blacklist technique. Furthermore, we define a new variable gained from the sender to record the amount of rejected IM messages, called: $\boldsymbol{Num_{reject}}$. The meaning of the rejected IM is the sender's IM messages blocked by global (or local) blacklist or reputation-based filtering method. Such information can be stored as part of the user's profile at server side and maintained as part of the state information when user logging on the server. By tracking such information at real-time, server can find out whose behavior is abnormal, i.e., whose Num_{reject} is higher than normal level or whose Num_{reject} is increasing very fast, etc. The users are classified into two types: *Spimmer* and *NormalUser*. We collect user's reputation value, Blacklist Level ($\mathbf{V_{black}}$) and Num_{reject} information to determine the user type.

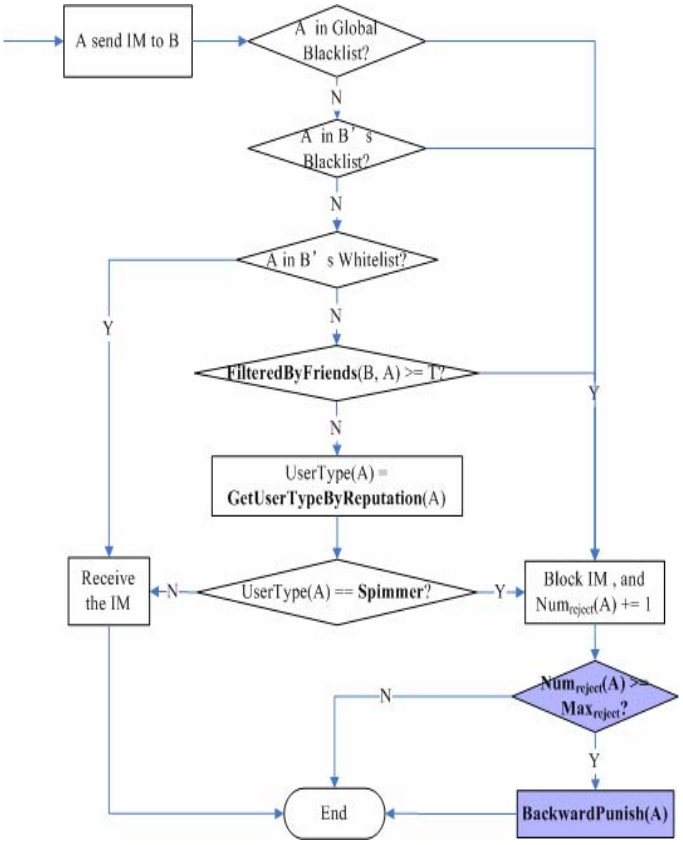

Figure 6. SpimRank filtering method

TABLE I. ALGORITHMS

procedure GetUserTypeByReputation (A)	**procedure** GetUserReputation (B, A)
var UserType := Spimmer	**var** reputation := **GetUserLocalReputation**(B, A)
var reputation := **GetUserReputation**(B, A)	**begin**
begin	**if** reputation < 0
if $\mathbf{V_{black}}$[A] == 0	reputation := **GetUserGlobalReputation**(A)
UserType := NormalUser	**end**
else if reputation > $\mathbf{Min_{rep}}$	**return** reputation
if V_{black}[A] <= $\mathbf{Min_{black}}$	
UserType :=NormalUser	
elseif Num_{reject}[A]<=$\mathbf{Min_{reject}}$	
UserType :=NormalUser	
end	**procedure** FilteredByFriends (B, A)
return UserType	**var** score := 0
	begin

```
procedure BackwardPunish(A)
begin
foreach user j in IM network
    if T_white[A][j] != 0
        Num_reject[j] += 1
end

Foreach user in B's contact-list
if user's blacklist contains A
    score := score +
            GetReputation(user)/Get
  AvgReputation()
end
return score
```

Some algorithms in Figure 6 is described in Table I. The variables $\mathbf{Min_{rep}}$, $\mathbf{Min_{black}}$ and $\mathbf{Min_{reject}}$ are threshold values that can be defined manually.

IV. Experimental Results

In this section we evaluate the effect of our filtering method in the five treat models. We simulate an IM network with 10,000 nodes, define variables and collect experimental results shown in Table II. The filtering method will be triggered when a spimmer sending spims to victims.

TABLE II. Teams of Variables Used in Our Experiment

Variables	Description
N	The amount of spimmers
N_{found}	The amount of detected spimmers
$N_{friends}$	the average amount of friends in user's contact-list
$N_{victims}$	The amount of victims that every spimmer will send spim to
P_{block}	The probability of adding spimmer to his blacklist when encountering with spim attack.
RemovedAttackers[i]	The amount of attackers removed after i-th round of attack
RemovedUsers	The amount of normal users who are missing justified as spimmer.
CurrentSpimCount[i]	The amount of spim messages the attackers may send in i-th round of attack
CurrentMissing-SpimCount[i]	The amount of spim messages received as normal IM messages in i-th round of attack

Threat Model A. *Individual Malicious Peers.* Malicious peers simply try to send spim to victims and doing nothing else.

We simulated a network with $N_{friends}$ = 25; $N_{attackers}$= 100. From experimental results shown in Figure 7, we can draw the conclusion that when N_{victim}* P_{block} ⩾ 2* Min_{reject} (Min_{reject} is the adjustable threshold values used by algorithm in Table I), the probability of finding out attackers can be raised to greater than 95%. For example, for the line of Min_{reject} = 5, when x = 10, we can find that y > 0.95. Similar result will be found for the lines of Min_{reject} = 10,15,20). It means that more than 95% of all attackers will be removed at the next round of attack (because they have been added to the victims' blacklist, and if they send spim to such victims again, their Num_{reject} will be increased to exceed the threshold Min_{reject}). Therefore, the higher N_{victim}* P_{block} is, the faster we can find out the attacker. It was indicated that 86% of the email users simply delete email spam instead of reporting spam to the server. In order to increase the P_{block} value, we should provide a client side option allowing automatically adding spim attacker to user's blacklist.

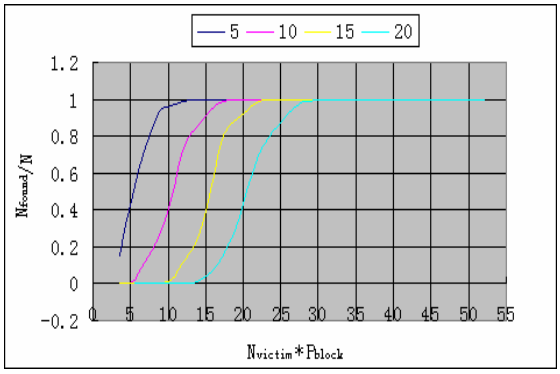

Figure 7. The experimental results of threat model A

Threat Model B. *Malicious Collectives.* Malicious peers know each other and unite to form a "malicious collective". In this collective, some of peers add other peers to their contact-lists in order to boost their reputation, then the peers with high reputation will send spim to victims. On another hand, some of the peers may impute some normal users by adding them into their blacklist, so that the Blacklist Level of these normal users will be increased.

We simulated a network with $N_{friends}$ = 25; P_{block} = 0.5; N_{victim} = 60; $N_{attackers}$= 100. These attackers form a malicious collective and 30 of them are boosted by others in the collective. Meanwhile, 50 of the normal users are imputed by peers in malicious collective and their Blacklist Level values are very high. The experimental result shows that *RemovedUsers* is always 0. Even though the *Blacklist Level* is high, reputation-based filtering method described in Table I also check the reputation and Num_{reject} values after examine the *Blacklist Level*. On another hand, the effect for attackers to boost peers in their collective is zero. It's because the PageRank-based reputation algorithm is a flow model based algorithm. Even though the malicious collective is large, it is isolated by the normal user's community. Normal users will not informally add such malicious peers into their contact-list. Therefore, the reputation flow will not transit into malicious collective and such boosting attack will be useless except normal user's community has links to the malicious collective. It may happen under the condition that there are malicious spies or zombie users in the normal user community, which will be discussed in threat mode D and E.

Threat Model C. *One time used identity attack.* Malicious peers always send spim to victims like threat model A, but only use such address once.

At the first attack, all spim are considered as normal IM messages. So without the algorithm *FilteredByFriends* (described in Table I), our method can not deal with threat model C. In the experiment we choose $N_{friends}$ = 25; $N_{attackers}$= 100; P_{block}= 0.5; and change parameter N_{victim} in every cases and record experimental results. Figure 8 shows the improvements of *FilteredByFriends*: the more victim spimmer attacks, the more effective.

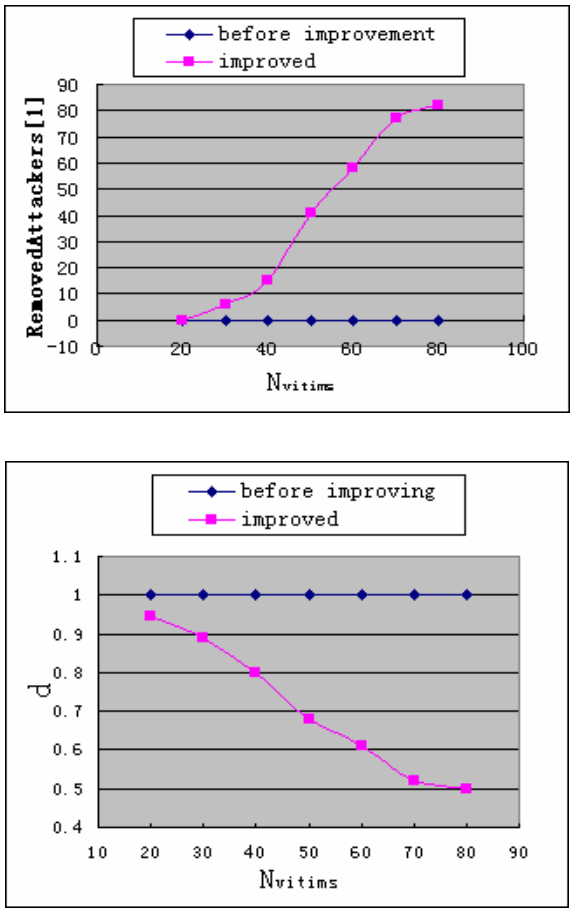

Figure 8. The experiment results of treat model C (*d*=CurrentMissingSpimCount[1]/CurrentSpimCount[1])

Threat Model D. Malicious peers in this model will use some normal users who are infected by virus or worms as zombie users, so that these victims may send spim or even boost these malicious peers to increase their reputation value.

Threat Model E. Malicious peers in this model may act like a normal users (like spies) and get high reputation from normal social network. But they use their high reputation to boost other peers who send spim.

In the above two threat models, both of attackers act as normal users, but add malicious peers into their contact-list to boost them. Therefore, we combine the two treat models into one experiment to find an effective way to detect such attackers.

When an attacker detected by our filtering method whose Num_{reject} is higher than our threshold Max_{reject}, the filtering approach *BackwardPunish* described in Table I will be triggered: the ones who add attacker into their contact-lists will be punished by increasing their Num_{reject}.

We choose $N_{friends} = 25$; $N_{attackers} = 150$; $P_{block} = 0.5$; $N_{victims} = \{20, 50\}$ and randomly select 50 peers from normal user's community to be malicious spies (or can be regarded as virus-infected users) who randomly add attackers to their contact-list. The experiment results are shown in Table III and we can see all the 50 spies can be detected.

TABLE III. THE EXPERIMENT RESULTS FOR THREAT MODEL D AND E.

The i-th round of attack		1	2	3
$N_{victim} = 20$	RemovedAttackers[i]	0	54	150
	the amount of spies detected	0	41	9
$N_{victim} = 50$	RemovedAttackers[i]	49	150	--
	the amount spies detected	50	--	--

V. CONCLUSIONS AND FUTURE WORK

With the fast deployment of IM applications, the spim attack has drawn serious public attention. Although there is similarity between spim and email spam, most of the anti-spam techniques cannot be perfectly applied to spim prevention, because of the real-time characteristic of IM.

In this paper, we propose a new anti-SPIM method *SpimRank*, which is based on trust and reputation mechanisms, and integrated with black-list/white-list techniques. We also make use of some unique mechanism in IM, such as maintaining session information for users logged on, and define a variable to record the user's historic action, which can be used to track user's abnormal behavior in order to detect spim attack as soon as possible. The *SpimRank* has the following advantages:

1 Detecting spim attacks in nearly real-time. In *SpimRank* filtering method, we can upgrade the blacklist level and Num_{reject} record immediately and servers always track such information in real-time. As a result, it is feasible to detect attackers' abnormal behaviors as soon as possible.

2 Light-weight. The filtering method makes full use of the IM characteristics, such as session information maintaining, contact-list and black-list mechanisms, etc. Therefore, we do not need to change much to the original IM service. The design to filtering method is based on integrating trust and reputation mechanisms with black-list/white-list techniques, which is simple but highly effective.

3 Highly resilient to attacks. The trust and reputation algorithm in SpimRank is based on a power iteration algorithm, which is typically highly resistant against attacks. Our experimental results show that the most of threat models can be resisted. Moreover, the filtering method can detect malicious spies and virus-infected users, which can't be detected by most of other methods.

4 Scalability. Power iteration algorithms have been improved to be computationally feasible even for very large graphs.

In the future work, we intend to enhance the algorithms and move from a centralized reputation system to a distributed system, in order to make the reputation system scalable for a large and multi-domains environment.

REFERENCES

[1] T. Thomas, "Spim, Like Spam, Is on the Rise". Information Week, 2004
[2] Z.J. Liu, et. al., "Detecting and Filtering Instant Messaging Spam - A Global and Personalized Approach", IEEE ICNP NPSEC, Nov. 2005.
[3] L. Page, et. al., "The PageRank Citation Ranking: Bringing Order to the Web", Stanford Digital Library Technologies Project Report, 1998
[4] A. Josang, R. Ismail, and C. Boyd, "A Survey of Trust and Reputation Systems for Online Service Provision." Decision Support Systems 2005. Available: http://dx.doi.org/10.1016/j.dss.2005.05.019