

# Toward Worm Detection in Online Social Networks

Wei Xu  
Pennsylvania State University  
University Park, Pennsylvania  
wxx104@cse.psu.edu

Fangfang Zhang  
Pennsylvania State University  
University Park, Pennsylvania  
fuz104@cse.psu.edu

Sencun Zhu  
Pennsylvania State University  
University Park, Pennsylvania  
szhu@cse.psu.edu

## ABSTRACT

Worms propagating in online social networking (OSN) websites have become a major security threat to both the websites and their users in recent years. Since these worms exhibit unique propagation vectors, existing Internet worm detection mechanisms cannot be applied to them. In this work, we propose an early warning OSN worms detection system, which leverages both the propagation characteristics of these worms and the topological properties of online social networks. Our system can effectively monitor the entire social graph by keeping only a small number of user accounts under surveillance. Moreover, the system applies a two-level correlation scheme to reduce the noise from normal user communications such that infected user accounts can be identified with a higher accuracy. Our evaluation on the real social graph data obtained from Flickr indicates that by monitoring five hundreds users out of 1.8 million users, the proposed detection system can detect the burst of an OSN worm when less than 0.13% of total user accounts are infected. Besides, by adopting simple countermeasures, the detection system is also shown to be very helpful for worm containment.

## Categories and Subject Descriptors

K.6.5 [Management of Computing and Information Systems]: Security and Protection

## General Terms

Security

## Keywords

worm, online social networks, early warning detection

## 1. INTRODUCTION

The popularity of online social networking (OSN) websites has been boosted in recent years. For example, Facebook has

more than 400 million active users [4] worldwide, MySpace has more than 70 million members [9] in U.S., Twitter has more than 75 million users [13] and LinkedIn has more than 44 million members [7]. This rapid growth of OSN websites has given rise to a number of security attacks. A representative example is a worm named Koobface, which spread in Facebook and MySpace in August 2008 [6] [10] [14]. It has generated 56 variants and has infected many other websites such as Tagged, Friendster, MyYearBook, Fubar.com, Hi5 and Bebo [1] [3] [8]. Besides Koobface, there are other worms, such as Mikeyy worm [12] and Samy worm [24] that have also caused havoc in OSN websites.

OSN websites have become an attractive target for these worms (hereinafter referred to as *OSN worms*) because of the following properties of the online social networks. First, online social networks are small-world networks [21], which means they have the properties of small average shortest path length and high clustering. The small average shortest path length property can reduce the propagation time from one user account to another user account. Meanwhile, the high clustering property suggests that users are tightly connected together, which facilitates the explosion of OSN worms. Second, online social networks are also scale-free networks [25], which are a class of power-law networks where high-degree nodes tend to connect with other high-degree nodes. When an OSN worm infects the account of a popular user (i.e., user with a large number of friends), this scale-free property suggests that the worm can infect another popular account shortly. As a result, the worm can achieve exponential growth by propagating to the large friends set of these popular users. Moreover, OSN worms also leverage social engineering [6] to increase the authenticity of worm messages.

By exploiting these properties of online social networks, OSN worms exhibit fast spreading characteristics similar to the ones observed in Internet worms. However, existing Internet worm detection mechanisms can not be directly applied to detecting OSN worms. This is because Internet worm detection heavily relies on the unique patterns of worm scanning traffic or the misbehavior of infected hosts, but neither of them can be observed in the propagation of an OSN worm. From the perspective of OSN websites (i.e., the server side), an infected user account does nothing but sending messages or posting updates as normal users do when the actual infection is taking place in the browser (i.e., the client side). This makes the detection of OSN worms a new and interesting problem.

In this paper, we propose an early warning OSN worm

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ACSAC '10 Dec. 6-10, 2010, Austin, Texas USA

Copyright 2010 ACM 978-1-4503-0133-6/10/12 ...\$10.00.

detection system. Early warning is essential for OSN worm detection because it provides administrators the opportunity to apply worm containment and elimination measures without affecting a large portion of user accounts. Meanwhile, achieving early warning in the detection of fast spreading worms is also a challenging problem [20].

Our approach leverages the properties of online social networks and the inherent propagation characteristics of OSN worms. More specifically, we first build a surveillance network based on the small-world and scale-free properties of the social graph to collect suspicious worm propagation evidence. We maximize the surveillance coverage by monitoring only a small fraction of user accounts. These accounts are selected by investigating the vertex properties of the social graph. We also realize that detection based on suspicious evidence alone is prone to high false positives, because worm activities are highly likely to be drowned out in normal user activities. As such, we further propose a scheme to effectively filter out the noise in the surveillance network. The implementation of the system is flexible, it can either reside on a dedicated server or be distributed within the OSN websites.

Our experimental results based on a real-life social graph dataset consisting of over 1.8 million users and 22.6 million friend links demonstrated the effectiveness of our detection system. The outbreak of an OSN worm can be detected when only 0.13% of the total user accounts are infected. Moreover, our detection system is simple in design, like a lightweight network intrusion detection system (NIDS), and it only needs to process a relatively small number of (suspicious) propagation evidence collected from the users in an EDS. We believe this could be a practical and easy solution for OSN websites that are currently struggled with OSN worms.

The rest of the paper is organized as follows: Section 2 provides the background on various OSN worms and characterizes their propagation vectors. We describe the design of the proposed detection system in Section 3. In Section 4, we evaluate its effectiveness in worm detection. We discuss several limitations of our work in Section 5 and describe some related works in Section 6. Finally, we conclude our work in Section 7.

## 2. BACKGROUND

Various OSN worms spread themselves by exploiting different features of the OSN websites. Nonetheless, their propagation vectors share certain similarities, which will be characterized by examining two representative OSN worms: Koobface and Mikeyy.

Figure 1 illustrates the propagation flow of Koobface in Facebook. User *A* receives a worm message from one of her friends (step 1) after this friend was infected by Koobface. Within this worm message, there is a link to a video clip hosted on a fake YouTube website. When user *A* clicks that link, the browser is redirected to the fake YouTube webpage (step 2), where the user is prompted by a request to install an update for “Adobe Flash player” plugin, which is actually a malware. After user *A* installs the claimed browser plugin (step 3), Koobface infects user *A*’s Facebook account and iterate its infection cycle by sending similar worm messages to all the friends in user *A*’s profile (step 4). Actually, besides sending messages, Koobface also sends invitations or composes posts, both of which contain similar worm content.

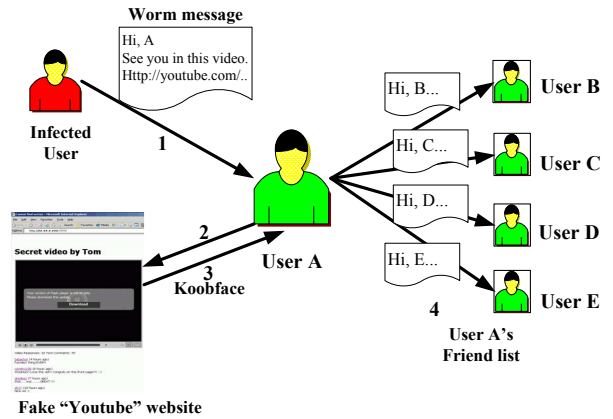


Figure 1: Koobface worm infection cycle

Mikeyy worm propagates by posting updates on an infected user’s profile to encourage the “follower” (people who can automatically receive the updates in their profiles) to visit [www.StalkDaily.com](http://www.StalkDaily.com), which was owned by the attacker. When a follower who is interested in the update clicks to see the poster’s profile, a self-replicated JavaScript code is injected into the follower’s profile. After the injection, similar updates are posted on the infected follower’s profile to repeat the infection cycle.

Despite the differences in the infection vectors of these two worms (E.g., downloading malware versus self-replicating JS code), both of them propagate following the social connections (E.g., friends or followers) of an infected account; in other words, their propagations follow the topology of the online social network (i.e., the social graph). One reason of this similarity is that social connections provide worms an opportunity to exploit social engineering such that the click-through rate of the malicious content can be increased. Besides, as mentioned before, topological properties of online social networks (e.g., small-world) can facilitate the spread of worms. Another similarity shared by both worms is the generation of passively noticeable activities such as worm messages and worm updates. This is because OSN worms are normally generated with certain malicious purposes such as advertising malicious websites or distributing malware.

In this paper, we limit our discussion to OSN worms that exhibit the above two properties. That is, we aim to detect worms that *propagate following the social connections* and *generate passively noticeable worm activities*. We acknowledge that not all existing OSN worms exhibit both the above properties and future OSN worms could take different formats. For example, Samy worm [2], a cross-site scripting (XSS) worm on MySpace, is one which does not generate activities passively noticeable by friends, so it is out of the scope of this work. We believe it would be better addressed by solutions focused on XSS vulnerabilities [24]. Indeed, the vulnerability exploited by Samy worm has been fixed and hence Samy worm does not work now, whereas the OSN worms we are addressing here remain a big threat to OSN sites.

## 3. SYSTEM DESIGN

In this section, we elaborate the design of the OSN worm

detection system, starting with a system overview.

As suggested by the high clustering property, the neighborhoods (i.e., friends) of most user accounts are densely connected. Therefore, for each neighborhood, our system only needs to monitor the “popular” user (i.e., the one with most friends in a neighborhood) to cover the entire neighborhood. Meanwhile, the scale-free property implies that a user with a large friend set tends to be friends with other users with large friend sets. This indicates that not all the popular users need to be monitored. Indeed, our system will only select a few of such users to maintain the surveillance coverage.

### 3.1 Overview

The general idea of our detection system is to deploy a disguised surveillance network being part of the online social networks to collect worm propagation evidence and to identify worm infections. Figure 2 illustrates the framework of our detection system, which consists of four major components. The *configuration module* retrieves from the administrator of the OSN website the social graph, based on which it determines where to collect evidence. The *evidence collecting module* gathers suspicious worm propagation evidence observed in an OSN website. The *worm detection module* identifies and reports a worm infection based on the input from the evidence collecting module. When an infection is detected, this module passes an alarm together with the infection information to the administrator of the OSN website via the communication module. The *communication module* provides all the necessary communications between an OSN website and the other modules. We will explain the design details of these modules in the following subsections.

One noteworthy property of our design is that each module only represents a combination of certain functionalities, and these functionalities can be implemented either within a dedicated server or in a distributed way. This property extends the flexibility of the system implementation, and it will be further discussed in Section 4.

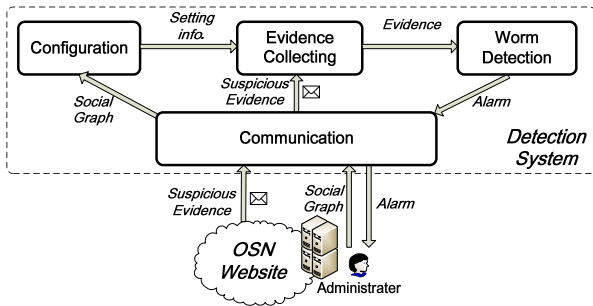


Figure 2: Detection System Overview

### 3.2 Evidence Collecting Module

The evidence collecting module is in charge of gathering worm propagation evidence (e.g., worm messages, worm updates). However, given the huge amount of information exchanged in an OSN website, the challenge is how to collect only suspicious worm evidence. Since OSN worms follow the social connections in propagation, a friend of an infected user account is more likely to receive worm propagation evidence.

To leverage this advantage, we adapt the idea of honeypot here as “decoy friend”. A decoy friend is a low-interactive honeypot, and it is created and added into a normal user’s friends list by the detection system. When a user account is infected by an OSN worm, decoy friends of that account can receive worm evidence. Similar ideas have been suggested in [30, 16] for other types of networks. In [30], the authors assume decoys only receive malicious messages. However, the same assumption does not hold in our work. In fact, our system treats the collected information from decoys only as suspicious evidence because some normal user activities can also be observed by decoys.

Decoys form a disguised surveillance network. We assign each decoy to be friends with several normal users so that a decoy can not be easily spotted because of its small number of friends. In addition, there are a few practical concerns regarding applying decoy friends in real world OSN websites. The first potential concern is related to user’s information privacy because decoys collect suspicious information in the network. However, since users’ data are all stored and kept in the OSN websites, we think our system will not cause new data/information leakage. Nevertheless, to alleviate such possible concern, our system will only keep the suspicious information for a short period of time. The second concern is that users might be reluctant to accept decoy friends. As such, a website will need to consult its users before assigning decoy friends to them. In fact, the OSN websites could provide incentives to encourage users to accept decoy friends. After all, both users and the OSN websites try to avoid worm infections for their own benefits. The third concern is on the number of decoy friends to be deployed in an OSN website. Besides user’s reluctance, the population of decoys may negatively impact the popularity of an OSN website, because decoy friends do not contribute to any interactive activities such as discussions or communications. To this end, *our system strives to limit the number of decoy friends while preserving the detection effectiveness*. We will discuss this design issue in the next section.

### 3.3 Configuration Module

The most important function of the configuration module is deploying decoys, which consists of two consecutive steps: selecting normal users and assigning decoy friends to these users. This module also performs other functions such as maintaining the configuration information of the system.

#### 3.3.1 Selecting Normal Users

Because of the practical concerns on applying decoys, our system only selects a small set of users (hereinafter referred to as “selected users set”) to be friends with decoys. Meanwhile, the objective of early warning favors a sufficiently large portion of users being kept under the surveillance of our system. Hence, the question is how to choose as small as possible a selected users set to achieve early warning. We formalize this problem in the context of social graph: Given a directed graph  $G = (V, E)$ , where each vertex denotes a user in the social network and each edge represents a connection between two users<sup>1</sup>, choose a minimum set of vertices such that each vertex either belongs to the set or there ex-

<sup>1</sup>in the case of mutual acquaintance between two users, such as in Facebook if  $A$  is friends with  $B$ , then  $B$  is also friends with  $A$ , this connection needs to be represented by two directed edges.

ists a path that ends at this vertex and starts from some vertex within the set. The length of this path is at most  $r$  hops. This problem is also known as *extended dominating set problem* [29] [19], which is NP-complete.

The choice of  $r$  affects the size of the selected users set. Therefore, it is carefully reasoned based on the following study of a real world social graph. Our study first confirms that given the same number of users, a larger  $r$  can cover a larger portion of the same social graph, so a larger  $r$  is desirable if at all possible. We also find that a worm starts from a single user can infect at most 0.08% of all users in two-hop propagation and 0.26% of all users in three-hop propagation. If our system sets  $r = 3$ , it is very likely that by the time worm propagation is detected, 0.26% of all users have been infected. This exceeds the early warning criterion (0.19%) suggested in [20], so our system sets  $r = 2$  as the coverage radius. Our study also suggests that it is not necessary to cover the entire social graph, because degree distribution of the vertices in a social graph follows power law distribution [25]. This property indicates that many vertices do not even have any connections. For example, over 20% of users in our evaluation data have no connections. These vertices are very unlikely to become the victim of a worm, and the effort of covering such vertices would produce a set with the size comparable to the size of the graph. Based on these studies of the properties of a social graph, we relax the constraint about covering the entire graph and redefine the problem as follow:

*Maximum Coverage Problem:* Given a social graph  $G = (V, E)$  and a number  $k$ , choose a set of vertices with size of at most  $k$  such that the number of other vertices that are covered by this set with coverage radius  $r = 2$  reaches the maximum.

The maximum coverage problem is also NP-complete. The previous extended dominating set problem reduces to it. Since both the scale-free property and the power-law distribution of degree suggest that high-degree vertices are more likely to be infected by a worm than most other low-degree vertices, these high degree vertices should be included in the selected users set with high priority. Besides, research on modelling epidemics in topological networks [15] [26] [31] also indicates that the more edges a node has, with a higher probability it will be infected quickly by an epidemic. These results suggest the following greedy heuristic: At each step, we add one vertex into the set such that the intersection between this vertex’s 2-hop coverage and the remaining vertices is maximum. Based on this heuristic, we design the following approximate algorithm.

The time complexity of Algorithm 1 is  $O(knm^2)$  where  $n = |V|$  and  $m = |E|$ . In practice, our system pre-processes the social graph to reduce the size of the graph such that the performance of the algorithm could be improved. Besides, since social graphs grow with time, we may run this algorithm periodically (e.g., once a week) to reflect such growth.

### 3.3.2 Assigning Decoy Friends

After a candidate selected users set is chosen, the configuration module sends the set to the OSN administrator, who will contact these users (with incentives) and return the final set of users that are willing to accept decoy friends. Upon receiving the final set, this module creates decoy profiles in

---

#### Algorithm 1 Maximum Coverage Algorithm

---

**Input:** Graph  $G = (V, E)$   
**Output:** Monitored user set  $C$

- 1:  $C \leftarrow \emptyset$
- 2: **while**  $|C| < k$  and  $V \neq \emptyset$  **do**
- 3:    $maxcover \leftarrow 0$
- 4:   **for**  $\forall v \in V$  **do**
- 5:      $cover_v \leftarrow$  2-hops coverage of  $v$
- 6:     **if**  $maxcover < cover_v$  **then**
- 7:        $maxcover \leftarrow cover_v$
- 8:     **end if**
- 9:   **end for**
- 10:  $C \leftarrow C \cup \{v\}$
- 11:  $V \leftarrow V - \{v\}$
- 12: **for**  $\forall u \in V$  and  $(v, u) \in E$  **do**
- 13:   **for**  $\forall (u, w) \in E$  **do**
- 14:      $V \leftarrow V - \{w\}$
- 15:   **end for**
- 16:    $V \leftarrow V - \{u\}$
- 17: **end for**
- 18:   Update degree of each  $v \in V$
- 19: **end while**

---

the OSN website and associates two decoy friends to each user by adding them into the user’s friends list (The justification of this scheme is discussed in Section 3.4). This module also modifies the account preference of each decoy friend (according to the setting of the OSN website) so that information received by decoy friends can be collected by the evidence collecting module.

## 3.4 Worm Detection Module

This module identifies the infected user accounts based on the suspicious worm propagation evidence. To distinguish actual worm evidence from normal user communications, this module applies correlation test on the suspicious evidence. The correlation test is based on similarities in the content and the structure of worm propagation evidence. One reason behind this similarity is that worm messages or updates composed by the same worm usually serve the same purpose (e.g., advertising a malicious link). Another reason is that the automatic message generation algorithms run by worms tend to reuse words and phrases because of the limited size of their candidate words set.

In this module, we employ a two-level spatial-correlation scheme, namely *local correlation* and *network correlation*. To provide necessary information to correlations, our system maintains a data structure called suspicious propagation evidence list (SPEL), which is associated with each selected user. In SPEL, every piece of evidence is stored as a  $\{decoy\ friend\ ID, receiving\ time, content\}$  tuple.

**Local Correlation:** Local correlation performs similarity test among suspicious evidence collected by two decoy friends assigned to the same selected user. The purpose of associating two decoy friends with one user is to offer a local reference such that upon receiving any evidence from one decoy friend, the system can search the other decoy friend’s SPEL for similar evidence. One of the following scenarios will happen:

1. Only one of the two decoy friends has received this message. With a high probability, this is worm prop-

agation evidence because a normal user is unlikely to send messages to one of his/her decoy friends, especially given that he/she knows which is a decoy.

2. Both decoy friends have received *similar but not identical* messages. With a high probability, this is worm propagation evidence because only the infected users send customized worm messages to each friend.
3. Both decoy friends receive the same message or update<sup>2</sup>. It could be either a group message or a worm message with the same content. In this case, the scheme resorts to *network correlation* for further identification.

**Network Correlation:** Network correlation is performed with input from all decoy friends. Upon receiving the same evidence from two decoy friends of a user, network correlation searches for similar evidence by computing the similarity score between the received evidence and any other evidence in the SPELs of other decoys. If similar evidence (e.g., with a similarity over 90%) is found, with high probability, both pieces of evidence can be confirmed as worm propagation evidence. We realize that some normal communications among users may have the similar propagation pattern as worm messages. For example, the outbreak of a large-scale event may cause similar or same messages distributed within an OSN. We will discuss this case in Section 6.

To examine the *similarity* between two pieces of suspicious propagation evidence, our scheme applies a simple measurement of similarity based on the metric of edit distance  $editDist()$  [23]. By this measurement, the *similarity* between evidence  $E_a$  and evidence  $E_b$  can be evaluated as follow:

$$sim(E_a, E_b) = \frac{1}{1 + editDist(E_a, E_b)} \quad (1)$$

where  $editDist()$  follows the definition of Levenshtein edit distance [22]. We acknowledge that more complex similarity measurements may generate more accurate results, but since that is not the focus of this paper, we will consider the other metrics in our future work.

Figure 3 gives an example illustrating the mechanism of two-level correlation. In this example, users  $A$ ,  $B$ ,  $C$  belong to the selected users set, and each of them is associated with two decoy friends. We assume that the worm first infects  $D$ , then it infects both  $E$  and  $A$ . After that,  $B$  is infected as well. At the time  $A$  was infected, if the worm sends customized messages to  $A$ 's decoy friends  $A1$  and  $A2$ , the system can identify  $A$ 's infection by local correlation between  $A1$  and  $A2$  (Scenario 2). If both  $A1$  and  $A2$  receive the same message (or update), the system checks the SPELs of all other selected users (e.g.,  $B$  and  $C$ ) and compares the received suspicious evidence with stored evidence. Because neither  $B$  or  $C$  is infected at this time, no match is found. Therefore, the evidence received by both  $A1$  and  $A2$  is stored in the SPEL of  $A$ . After  $B$  is infected, the same procedure is preformed and a match can be found between the evidence stored in the SPEL of  $A$  and the evidence just received by  $B1$  and  $B2$ . At this time, the infection of both  $A$  and  $B$  can be identified.

<sup>2</sup>Since updates are automatically displayed to all the friends by the OSN website, updates cannot appear in the previous scenarios

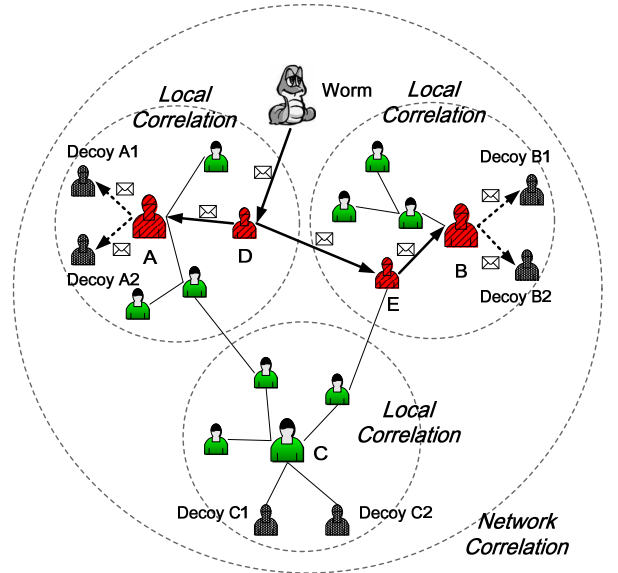


Figure 3: An example of two level correlation

### 3.5 Communication Module

The communication module acts as the interface of the detection system since it processes all necessary communications between the system and the OSN website. For example, it coordinates the communications between the configuration module with the administrator during system setting up. It receives propagation evidence from the OSN websites and passes them to the evidence collecting module. It also sends the worm infection alarm to the administrator on behalf of the worm detection module.

## 4. SYSTEM EVALUATION

In this section, we evaluate the OSN worm detection system on the real world social graph of Flickr [5]. There are 1,846,198 users and 22,613,981 friend links in the social graph, and the average friend number per user is 12.24. The data set is crawled from Flickr for a measurement study [25].

We evaluate the detection system with three objectives. The first objective is early warning. We define early warning in terms of the number of infected accounts by the time the worm is detected. This metric is borrowed from Internet worm detection. As suggested in [20], a worm detection system is deemed as an early warning system if the worm propagation is detected when less than 0.19% of all vulnerable hosts are infected. The second objective is to test the detection system under various worm propagation behaviors as well as under practical constraints such as user reluctance. The third objective is to examine the effectiveness of our system in worm containment with simple countermeasures provided.

To assess the practicability of the system design, we also discuss the implementation of the detection system in real world OSN websites.

### 4.1 Simulation Model

Our simulation model consists of four modules as shown in Figure 4. The initialization module takes the social graph

as input and output the set of selected users (the default size of the set is 500). For each selected user, this module adds two edges in the social graph from the vertex (representing the user) to the two new vertices (representing two decoy friends). It also creates and associates a SPEL with each selected user. Worm propagation starts from certain user(s). The propagation module models worm propagation by repeating two consecutive phases, namely sending worm messages (both messages and updates are referred as messages in evaluation) and infecting users. In the process of sending worm messages, our simulator chooses either to send worm messages to all the friends or to a fraction of friends randomly chosen from the friends list of an infected user. Each worm message is randomly selected from the a predefined worm messages set. We assume each recipient of worm messages gets infected with a probability of  $P_{user}$ . The probability is randomly assigned for each user and keeps constant in each worm propagation. The monitor module performs correlation tests on suspicious worm messages. Once a worm infection is identified, the post-processing module can output the infection statistic such as percentage of infected users.

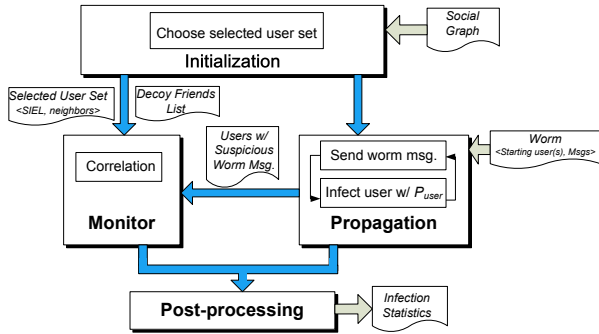


Figure 4: OSN Worm Simulation Model

Since a few random variables are used in each propagation, we repeat each propagation for 100 times in our evaluation to reduce the impact of randomness. After that, we display the results as the mean values of these iterations as well as 95% confidence intervals of the means.

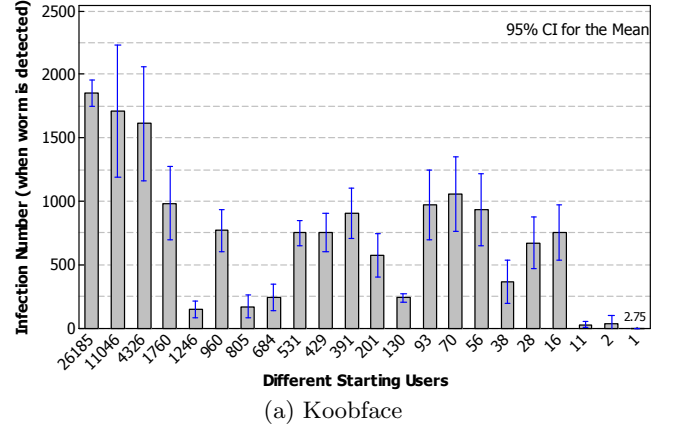
## 4.2 Early Warning Detection

In this part of evaluation, we examine whether the detection system can achieve early warning. To this end, we test the detection system in the propagation of two representative OSN worms, namely Koobface worm and Mikeyy worm. A user account infected by Koobface worm sends different (customized) messages to its two decoy friends. In Mikeyy worm propagation, an infected account delivers the same message to both decoys.

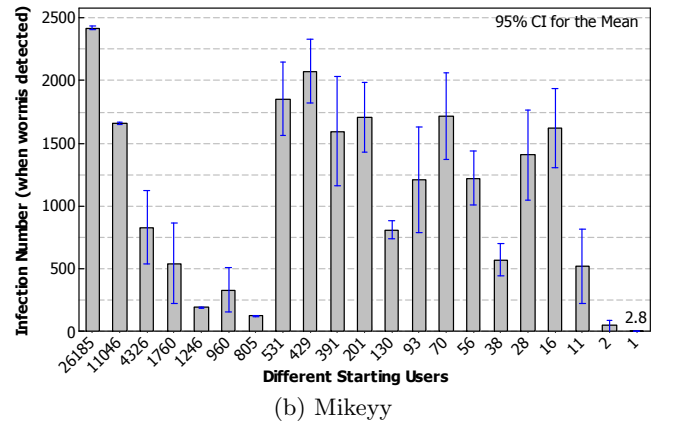
A crucial factor in worm propagation is the initially infected user account(s) (i.e., hitlist) because the total friend numbers of the initial account(s) can affect the worm propagation speed. To conduct a more comprehensive test, we choose 22 accounts from the Flickr dataset with different friend numbers (from 1 to 26,185, where 26,185 is the maximum number of friends) as initial infected user accounts and start worm propagation from these accounts in both worm cases. Then we measure the average infection numbers by

the time these worms are detected and the results are showed in Table 1 and Figure 5.

Table 1 lists the average infection numbers on detection for both worms. The maximum infection number is 2420, which is only 0.13% of all the users. This indicates the detection system fulfills the early warning requirement. In addition, Table 1 shows the average infection number of Mikeyy worm is larger than that of Koobface worm. This is because the detection of Mikeyy worm infections requires both local correlation and network correlation. Next, we will use the Mikeyy worm propagation model for evaluation.



(a) Koobface



(b) Mikeyy

Figure 5: Infection Number versus Different Initial User Accounts

Figure 5 confirms that in general propagations starting from “popular” users can infect more user accounts. However, we also notice that in some cases worm propagation starting from less popular initial users can infect more user accounts than the propagations starting from more popular initial users. For example, in both worms, propagations starting from the user with 531 friends infects more accounts than propagations from the user with 1246 friends. The reason is that popular users are more likely to be included in the selected users set. If a user  $A$ , who has more friends than another user  $B$ , has decoy friends, the infection of  $A$  will be detected faster than infection of  $B$ . Therefore, a worm starting from  $A$  may infect less accounts than the same worm starting from  $B$ . Another reason, as demonstrated in [25], is that social graph tends to exhibit a tightly-connected “core”

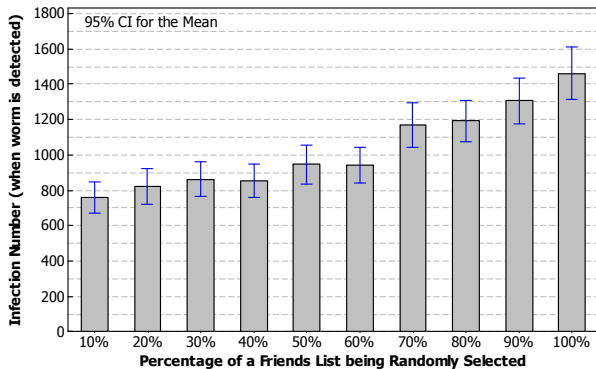
**Table 1: Infection Number on Detection for Koobface and Mikeyy Worms**

Worm Type	Avg. Infection #	Max Infection #	Min Infection #
Koobface	700	1851	2.75
Mikeyy	1023	2420	2.8

of high-degree vertices connected with each other because of its scale-free metric and the positive assortatively coefficient. This property implies that when a “popular” user is infected, even if it does not belong to the selected user set, it is very likely to infect another “popular” user in a short time and this infection can be detected because the other “popular” user has decoy friends. This observation also justifies the heuristic we adopt in Algorithm 1, where we start with high-degree vertices.

### 4.3 Impact of Worm Behavior

In this part, we evaluate the detection system under behavior discrepancy of OSN worms. More specifically, we consider that an OSN worm randomly chooses a fraction of friends of an infected user as its propagation targets. We note current OSN worms infect all friends, making them easier to detect. Our evaluation is to show the effectiveness of our scheme against more intelligent worms. The impact of this behavior discrepancy is that some decoy friends may not receive propagation evidence even if the user accounts to which they are attached have been infected. To test our detection system under this assumption, we simulate worm propagations with usage of friend lists from 10% to 100%. For each percentage, worm propagation starts from the above 22 different accounts and the average results are showed in Figure 6.

**Figure 6: Infection Number versus Different Percentages of Friends lists**

In Figure 6, all the worm propagations are detected when less than 1600 user accounts are infected. This indicates that the detection system can still achieve early warning when worms randomly choose targets from friend lists. We notice that even the worms using only 10% of the friend lists can still be detected. Hence, as long as worms have no knowledge about the identities of decoys, shrinking the size of target lists to reduce the probability of hitting decoys is ineffective to the attacker. On the other hand, if somehow worms spot some decoys, they can evade these known decoys (we will discuss the impact of this scenario in the next section). Another trend illustrated by Figure 6 is that

when more friends are targeted, more accounts can be infected by worms. Therefore, OSN worms will tend to use as many contacts in the friends lists as possible if their goal is to enlarge infection.

### 4.4 Impact of Selected Users Set

As mentioned previously, not all the users in the selected users set are willing to accept decoy friends. Besides, we also consider the scenario where some of the decoys are spotted so that worms can avoid these decoys in propagation. To evaluate the detection system in these scenarios, we study worm propagations where only a part (randomly chosen) of the selected users set has decoy friends. We first choose a selected users set of 2000 users by Algorithm 1. After that, we randomly choose 100 to 2000 users from this set to assign decoy friends and then run worm propagations for each case. Again, for each set, worm propagation starts from the previously mentioned 22 different user accounts and the average infection numbers by the time of detection are illustrated in Figure 7.

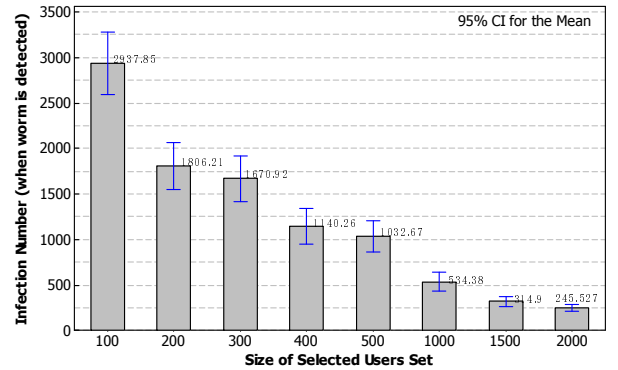
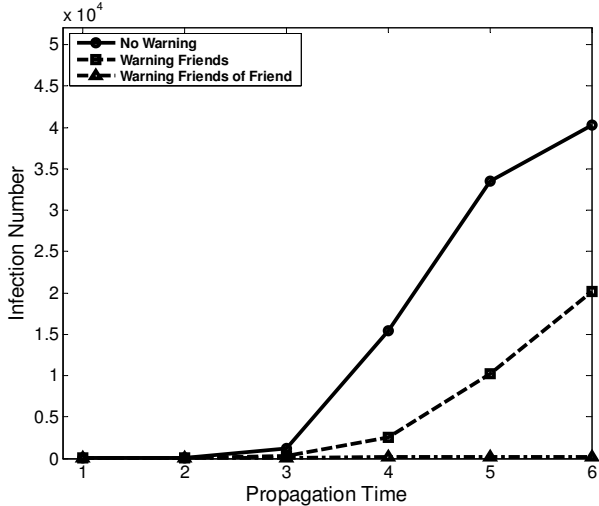
**Figure 7: Infection Number versus the Size of Selected Users Set**

Figure 7 clearly shows that a larger set of selected users with decoy friends can detect worms with fewer infected user accounts. For example, 1500 selected users set can detect worm propagations when only 314 user accounts are infected. However, when the size of the set is smaller or equal to 100, the infection number is larger than 3000, which implies that early warning may not be fulfilled with a selected users set at this size. On the other hand, the infection numbers are restrained less than 1000 when the set size is larger or equal to 500. This result shows the effectiveness of our detection system under the impact of a partial working surveillance network. It also indicates that the administrator of an OSN should encourage more users (if at all possible) to accept decoy friends, e.g., with incentives.

### 4.5 Containment Measures

Upon detecting a worm propagation, the system will notify the administrators of OSN websites. In addition, the

detection system can assist in suppressing worm propagations by adopting some simple countermeasures, such as warning the friends of infected users [11] (1-hop warning) by decoy friends or also warning the friends of friends (2-hops warning) if the privacy setting of a selected user allows the decoy friends to retrieve the friends information. In this study, we assume users will raise their vigilance after receiving the warning messages and for simplicity here we assume they will not be affected by the worm. The following results demonstrate the effectiveness of such warning mechanisms in worm containment.



**Figure 8: Worm Propagation versus Different Containment Measures**

Figure 8 compares the infection numbers with and without warnings (both 1-hop and 2-hops warnings). The results are based on a setting where a set of 500 selected users are used to detect and warn other users. Figure 8 illustrates that warnings can effectively suppress the worm propagation. A 2-hop warning approach can limit the infected user number to a small value compared with the case in which no warning is issued. A 1-hop warning prevents nearly half of the users from being infected compared with the no warning case. These results with the simple countermeasures (e.g., 1-hop and 2-hops warnings) indicate the detection system is helpful in worm containment.

#### 4.6 Example Implementation

We discuss a practical implementation of the worm detection system in detail based on a realworld OSN website, Facebook. The system can reside in a dedicated server. It has access to the Internet so that the server can visit Facebook and communicate with the administrator of Facebook through a secured channel (e.g., HTTPS or SSH). The modules run as programs on the server. All the modules have access to a database, which stores datasets such as the social graph of Facebook, selected users set, accounts of generated decoys and their login credentials, SPELs set, and identified worm evidence. The configuration module retrieves the social graph from the database and outputs selected users set to the database. It can also invoke an instance of a Web browser to register the decoy accounts on Facebook. The

evidence collecting module is composed of an email client and a Web browser. By configuring the preference of each decoy account in Facebook, this program will be notified via emails about any suspicious evidence received by decoys. Upon the arrival of a notification email, the program of collecting module logs into the decoy account through the Web browser, retrieves the suspicious evidence and writes the evidence into the corresponding user’s SPEL in the database. After that, the program invokes the worm detection program to process the evidence. The worm detection program can send notification to the administrator of Facebook via the secure channel if it identifies any infection based on the existing evidence in the database.

Although this implementation is based on Facebook, the features of Facebook used by the system (e.g., email notification) are widely supported by other OSN websites. Therefore, we believe this implementation can be adapted for other OSN websites with a slight modification. Moreover, our system can also be implemented in a distributed way. For example, the detection functionalities of the system may be distributed among decoys such that each decoy can perform its own worm detection by sharing suspicious evidence with other decoys.

## 5. LIMITATIONS

In this section, we discuss some limitations of our detection system.

One practical concern is raised from the scenario where normal messages spread in a worm-like pattern within an OSN website due to the outbreak of a large-scale event. For example, a breaking news can be broadcasted from one user to all his friends and keep multiplying in this way. There could be little difference between the propagation pattern of this news and an OSN worm. Unless there exists an approach to automatically distinguish the normal news from a worm message based on the content, little can we do to avoid the false alarms caused by this scenario. For example, if a posted link in a collected suspicious message is pointed to some well-known news websites, we may ignore it. Otherwise, which we believe is a very rare case, simple manual checking of the message content by an administrator will address the false alarm problem.

Another practical concern is in regards to the infection speed of OSN worms. Although one characteristic of OSN worms is fast infection, some of the worm infection cycles may be longer than the detection time window because users may respond to worm messages with different latencies. In this case, the only solution is to expand the time span of correlation. However, this could also degrade the performance of the detection system. As such, there is a trade-off between the computation resource and the demand of OSN websites.

## 6. RELATED WORK

In the area of Internet worm early detection, various detection strategies have been proposed. Gu *et al.* suggested an algorithm based on local victim information [20], in which they used destination-source correlation to capture the pattern in incoming and outgoing scanning traffics of a host before and after it is infected by a scan-based worm. They also looked for worm’s anomalous scanning patterns, such as high scan rate to identify the outbreak of a worm. However,

their approach does not apply to OSN worm detection because no such scan traffic are present. Dagon *et al.* proposed a detection technique [18] using honeypot to monitor the entire infection process (infection cycle) rather than just the beginning and the end. They recorded memory event, network event and disk event to perform logistic analysis looking for correlation. Their approach requires no signature in advance and has the advantage of coping with polymorphic worms. However, lack of infection processes in OSN worms prevent applying their approach here. In fact, due to very limited worm activities, any approaches relying on detailed infection procedure is not suitable here.

Bu *et al.* suggested a worm detection scheme [17] based on the extraction of the alteration of arrival unsolicited scan rates in the early stage of worm propagation. Their work suggested a novel signal indicating the outbreak of an Internet worm, but this approach suffers from the problem of too many potential sources for false positive rate. Wagner *et al.* provided an entropy based worm detection algorithm [27]. They utilized entropy to quantify the difference of randomness observed in worm traffic and in normal traffic. The source IP address fields will be less random in worm traffic than in normal traffic since the scanning hosts' IP address were seen more than other hosts. Their strategy offered an alternative way to detect the propagation activities of an Internet worm. However, both of these approaches rely on the characteristics exhibited in worms' scanning traffic. For an OSN worm, no scan is necessary and the infection traffics are relatively simple compared to that of internet worm. Unlike packets with various attributes transmitted during the propagation of Internet worms, OSN worms merely generate messages. Moreover, there is no hierarchical structure in the organization of a social network. All peers are equal in the social graph, which means no auxiliary information is available for decisions of the location of a worm detector.

There are some other worm detection algorithms that are not based on scanning traffic. Wang *et al.* proposed an anomalous payload-based worm detection algorithm [28], a worm propagation can be identified if correlation of ingress and egress payload alerts is observed. In an OSN worm, the actual payload is downloaded in the browser, which cannot be observed by OSN websites. This is actually exploited by OSN worms to bypass any filtering based detection scheme deployed in OSN websites. However, one thing we can borrow from this work is the idea of correlation. As showed in this paper, we adopt correlation in worm activities to improve the detection accuracy.

## 7. CONCLUSIONS

In this paper, we design a system that can effectively detect the propagation of OSN worms. By exploiting the properties of OSNs, we construct a surveillance network embedded in the OSN websites using decoy friends. We also propose an algorithm based on the heuristic derived from the topological properties of social graphs to keep the OSN websites under surveillance by monitoring only a few hundreds of users. We leverage both local and network correlations of worm propagation evidence in our detection system to achieve early warning detection.

Based on the real-world social graph of Flickr, our evaluation with two known worms, Koobface and Mikey, shows that the detection system can effectively detect OSN worm propagations when less than 0.13% of total user accounts are

infected. Even taking user reluctance into consideration, our system can still achieve early warning detection. Moreover, the detection system is also demonstrated to be applicable to worm containment by adopting some simple countermeasures. This can provide valuable assistance to OSN websites in fighting against worm propagations in future.

## 8. ACKNOWLEDGMENTS

We thank Alan Mislove for providing us with the social graph data set of Flickr.

## 9. REFERENCES

- [1] 56th variant of the koobface worm detected. <http://blogs.zdnet.com/security/?p=3414>.
- [2] Cross-site scripting worm floods myspace.
- [3] Facebook security threats. <http://www.facebook.com/security?v=app\4949752878&viewas=1661798617&ref=search>.
- [4] Facebook statistics. <http://www.facebook.com/press/info.php?statistics>.
- [5] Flickr. <http://www.flickr.com/>.
- [6] Koobface virus hits facebook. <http://news.cnet.com/koobface-virus-hits-facebook/>.
- [7] LinkedIn: About us. <http://press.linkedin.com/about>.
- [8] Msrt august top detection reports. <http://blogs.technet.com/mmpc/archive/2009/08/27/msrt-august-top-detection-reports.aspx>.
- [9] Myspace fact sheet. <http://www.myspace.com/pressroom?url=/fact+sheet/>.
- [10] New worms target both myspace and facebook users. <http://www.kaspersky.com/news?id=207575670>.
- [11] Steps you should take. <http://www.facebook.com/security>.
- [12] Teen claims responsibility for disrupting twitter.
- [13] Twitter users number.
- [14] W32.koobface.a. [http://www.symantec.com/security\\_response/writeup.jsp?docid=2008-080315-0217-99&tabid=2](http://www.symantec.com/security_response/writeup.jsp?docid=2008-080315-0217-99&tabid=2).
- [15] M. Boguna, R. Pastor-Satorras, and A. Vespignani. Epidemic spreading in complex networks with degree correlations. *Lecture Notes in Physics: Statistical Mechanics of Complex Networks*, 625:127–147, 2003.
- [16] G. Brown, T. Howe, M. Ihbe, A. Prakash, and K. Borders. Social networks and context-aware spam. In *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work*, San Diego, CA, 2008.
- [17] T. Bu, A. Chen, S. Vander Wiel, and T. Woo. Design and evaluation of a fast and robust worm detection algorithm. In *Proceedings of the 25th IEEE International Conference on Computer Communications*, Barcelona, Catalunya, Spain, 2006.
- [18] D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levine, and H. Owen. Honeystat: Local worm detection using honeypots. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection*, Sophia Antipolis, French Riviera, France, 2004.
- [19] Y. Fu, X. Wang, and S. Li. Construction k-dominating set with multiple relaying technique in wireless mobile

- ad hoc networks. In *Proceedings of the 2009 WRI International Conference on Communications and Mobile Computing*, Kunming, Yunnan, China, 2009.
- [20] G. Gu, M. Sharif, X. Qin, D. Dagon, W. Lee, and G. Riley. Worm detection, early warning and response based on local victim information. In *Proceedings of the 20th Annual Computer Security Applications Conference*, Tucson, AZ, 2004.
- [21] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA, 2006.
- [22] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics-Doklady*, 10:707–709, 1966.
- [23] D. Lin. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, Madison, WI, 1998.
- [24] B. Livshits and W. Cui. Spectator: Detection and containment of javascript worms. In *Proceedings of the 2008 USENIX Annual Technical Conference*, Boston, MA, 2008.
- [25] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, San Diego, CA, 2007.
- [26] Y. Moreno, R. Pastor-Satorras, and A. Vespignani. Epidemic outbreaks in complex heterogeneous networks. *The European Physical Journal B - Condensed Matter and Complex Systems*, 26:521–529, 2002.
- [27] A. Wagner and B. Plattner. Entropy based worm and anomaly detection in fast ip networks. In *Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, Linkoping, Sweden, 2005.
- [28] K. Wang, G. Cretu, and S. J. Stolfo. Anomalous payload-based worm detection and signature generation. In *Proceedings of the 8th International Symposium of Recent Advances in Intrusion Detection*, Seattle, WA, 2005.
- [29] J. Wu, M. Cardei, F. Dai, and S. Yang. Extended dominating set and its applications in ad hoc networks using cooperative communication. *IEEE Trans. Parallel Distrib. Syst.*, 17(8):851–864, 2006.
- [30] M. Xie, Z. Wu, and H. Wang. Honeyim: Fast detection and suppression of instant messaging malware in enterprise-like networks. In *Proceedings of the 23th Annual Computer Security Applications Conference*, Miami Beach, FL, 2007.
- [31] C. C. Zou, D. Towsley, and W. Gong. Modeling and simulation study of the propagation and defense of internet e-mail worms. *IEEE Trans. Dependable Secur. Comput.*, 4(2):105–118, 2007.