A Study of XSS Worm Propagation and Detection Mechanisms in Online Social Networks

Mohammad Reza Faghani, *Student Member, IEEE*, and Uyen Trang Nguyen, *Member, IEEE*

***Abstract*—We present analytical models and simulation results that characterize the impacts of the following factors on the propagation of cross-site scripting (XSS) worms in online social networks (OSNs): 1) user behaviors, namely, the probability of visiting a friend's profile versus a stranger's; 2) the highly clustered structure of communities; and 3) community sizes. Our analyses and simulation results show that the clustered structure of a community and users' tendency to visit their friends more often than strangers help slow down the propagation of XSS worms in OSNs. We then present a study of selective monitoring schemes that are more resource efficient than the exhaustive checking approach used by the Facebook detection system which monitors every possible read and write operation of every user in the network. The studied selective monitoring schemes take advantage of the characteristics of OSNs such as the highly clustered structure and short average distance to select only a subset of strategically placed users to monitor, thus minimizing resource usage while maximizing the monitoring coverage. We present simulation results to show the effectiveness of the studied selective monitoring schemes for XSS worm detection.**

***Index Terms*—Computer worms, cross-site scripting, malware, online social networks, worm propagation modeling.**

I. Introduction

ONLINE social networks such as Facebook, Twitter and MySpace have attracted hundreds of millions of people worldwide who use this service to connect and communicate with their friends, family and colleagues geographically distributed all around the world. This service cuts two ways, however. On one hand, OSNs are an ideal place for people to gather, communicate, socialize and share their common interests. On the other hand, malware creators often exploit the trust relationship among OSN users to propagate automated worms through online social networks. The first OSN worm, Samy, that hit MySpace in 2005 by exploiting a cross-site scripting (XSS) vulnerability in a MySpace web application infected about one million victims within 24 hours [1].

XSS worms exploit existing vulnerabilities in web applications to propagate themselves. An XSS worm usually infects members of an OSN in two steps. In the first step, the worm creator embeds the malicious code into his/her (usually fake) profile or wall. In the second step, any person who subsequently visits the infected profile will inadvertently execute the embedded malicious code. An XSS flaw (such as the one exploited by Samy) will help the worm to execute the malicious code in the visitor's browser while an AJAX (Asynchronous JavaScript and XML) technology unintentionally enables the code to embed itself into the visitor's profile. The visitor's profile then becomes infected, which allows the worm to propagate further in the OSN [1].

There has not been any in-depth research on XSS worms and their propagation in *online social networks*. In fact, the topic of malware in OSNs has only been studied recently. However, existing works focus on prevention, detection, containment and elimination of malware [2]–[7]. Our work in this article focuses on characteristics of malware propagation in OSNs, which will allow us to design more effective and resource-efficient countermeasures.

In several OSNs such as Facebook, LinkedIn, Orkut, and hi5, the relationship (friendship) between two users is mutual. Such an OSN can be represented by an undirected graph $G = (V, E)$ in which each vertex (or node) $v \in V$ represents a user, and an edge $e \in E$ between two vertices indicates the existence of a relationship (friendship) between the two respective users. In this article, we consider only OSNs that can be represented by undirected graphs.

OSNs have a common and distinctive property: the property of highly clustered communities [8]. That is, an OSN is made up of highly clustered communities with connections between members in a community being dense while connections between different communities are sparse. A strong definition of community requires all members of a community to be connected to each other, which leads to a definition of *clique*. A clique in the graph G is a subgraph $G' = (V', E')$ such that $V' \subseteq V$, $|V'| \geq 3$, $E' \subseteq E$ and, for every two vertices in V', there exists an edge connecting the two. A maximal clique is a clique that does not exist exclusively within the vertex set of a larger clique [9].

We identify three major factors that have significant effects on the XSS worm propagation speed (infection rate) in an OSN. They are (1) user behaviors, namely, the probability of visiting a friend's profile versus a stranger's; (2) the highly clustered structure of communities; and (3) community (clique) sizes. We present analytical models and simulation results that characterize the impacts of the above three factors on the XSS worm propagation speed. The proposed analytical models and simulation results show that the clustered structure of a community and users' tendency to visit their friends more often than strangers help slow down the propagation of XSS worms in OSNs.

Manuscript received October 30, 2012; revised February 18, 2013, May 14, 2013, and August 13, 2013; accepted August 19, 2013. Date of publication September 05, 2013; date of current version October 18, 2013. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Kui Ren.

The authors are with the Department of Computer Science and Engineering, York University, Toronto, ON, M3J 1P3, Canada.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2013.2280884

The obtained results motivated us to go one step further by identifying and evaluating potential algorithms for more resource-efficient detection mechanisms. Currently, it is a common practice by administrators of OSNs such as Facebook to performs real-time checking on every read and write post. That amounts to 25 billion posts checked per day, which reaches 650,000 posts checked per second at its peak [4]. Given a huge OSN such as Facebook, currently having about 900 million users and growing, this practice is not very efficient. Instead of this exhaustive checking method, a few methods based on selective monitoring have been proposed; some are for OSNs [2], [3], [5] while the others are for other types of networks [10]. Using these selective monitoring methods, we select only a set of important users in the network and monitor their and their friends' activities and posts for malware threats. These methods differ in how the set of important users is selected. In this article, we present a study of several selective monitoring schemes. In particular, we evaluate and compare their effectiveness in terms of malware detection in OSNs.

The remainder of the paper is organized as follows. In Section II, we describe the system model and simulation parameters used in the article. In Section III, we present an analytical model that characterizes XSS worm propagation in OSNs based on users' probability of visiting friends versus strangers. In Section IV, we study the impact of the clique size on the propagation speed. We continue our study of XSS worm propagating in Section V by presenting simulation results that demonstrate the effect of the clustering coefficient on malware propagation. In Section VI, we discuss and compare several selective monitoring schemes used for malware detection. Related work is presented in Section VII. We conclude the paper and outline our future work in Section VIII.

II. System Model and Simulation Parameters

We represent an OSN using an undirected graph $G = (V, E)$ in which each vertex (or node) $v \in V$ represents a user, and an edge $e \in E$ between two vertices indicates the existence of a relationship (friendship) between the two respective users. There exist many OSNs in which the relationship (friendship) between two users is mutual (e.g., Facebook, LinkedIn, Orkut), and they thus can be represented by undirected graphs.

An OSN has the following three distinct characteristics [11], [12] that make worm propagation different from that in other types of networks (e.g., computer networks).

1) A social network typically has a low average network distance, approximately equal to $\log N / \log d$, where $N = |V|$ is the number of vertices (people), and d is the average vertex degree of the graph G.
2) Node degrees of a social network graph tend to be or, at least approximately, power-law distributed.
3) Social networks typically show a high clustering property, or high local transitivity. That is, if person A knows B and C, then B and C are likely to know each other. Thus A, B and C form a *friendship triangle*. Let k denote the degree of a vertex v. Then the number of all possible triangles originated from vertex v is $k(k-1)/2$. Let f denote the number of friendship triangles of a vertex v in an OSN graph. Then the clustering coefficient $C(v)$ of vertex v is defined as $C(v) = 2f/(k(k-1))$. The *clustering coefficient* of the graph is the average of the clustering coefficients of all of its vertices. Clustering coefficients of real-life OSNs range from 0.1 to 0.7 [11], [12].

TABLE I
OSN Graph Used in Our Simulation

Parameter	Value	Value	Value	Value
Number of vertex (people)	3000	10,000	20,000	100,000
Number of edges	8991	29991	59991	299991
Average clustering coefficient	0.19	0.15	0.14	0.11
Average shortest path length	5.1	5.5	5.7	6.4
Network diameter	11	13	13	15
Maximum node degree	129	226	456	1281
Average node degree d	5.99	5.99	5.99	5.99
$\log(n)/\log(d)$	4.4	5.14	5.52	4.2

There exist a few algorithms that can generate social network graphs with the above characteristics [11], [13]. For the simulations reported in this paper, we use the algorithm proposed by Holme and Beom [11]. We generated four OSN graphs of sizes $N_1 = 3,000$, $N_2 = 10,000$, $N_3 = 20,000$, and $N_4 = 100,000$ nodes. The parameters and characteristics of the OSN graphs are listed in Table I.

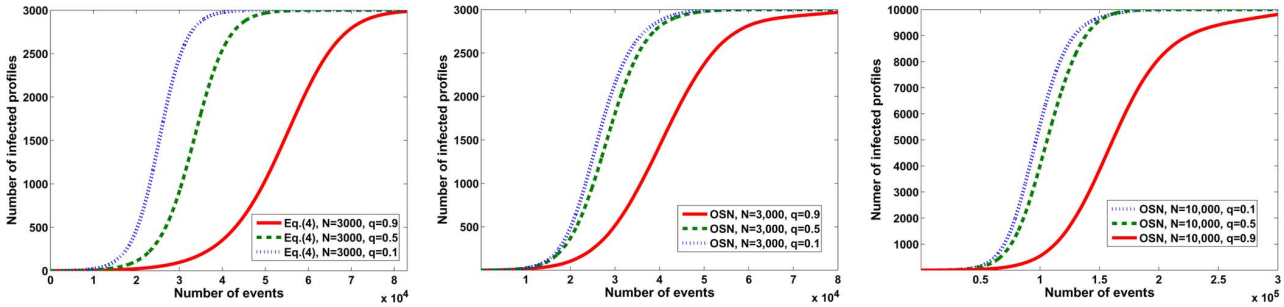
We define an *event* or a *visit* in an OSN to be the action of visiting (accessing) a user's profile by some other user. We assume that events in an OSN happen consecutively one after another. (Two different users may click on the same profile at the same time. Their access requests, however, will be queued at a server consecutively, waiting to be processed. The two events are thus considered to happen one after the other.)

The simulation software is implemented using MATLAB. The simulation is of discrete-event type, consisting of discrete virtual time slots. A time slot is equivalent to an *event* defined above. In each time slot, a user (node) j is chosen randomly with a probability $\phi = 1/N$ and the user will visit a friend's profile with a probability q_j and a nonfriend user's profile with probability $1 - q_j$. Two users are friends if and only if their corresponding vertices in the OSN graph is connected by an edge $e \in E$. Each data point in the result graphs is the average of 100 runs, each with a different random seed.

If a user's browser has add-on protections (e.g., NoScript add-on for Firefox browsers) to prevent XSS scripts from running automatically, that user is considered not vulnerable to XSS worms. We will consider only *vulnerable users* in our analysis and simulations.

III. User Behaviors

In the case of XSS malware, user behaviors can be characterized by the tendency of visiting friends' profiles versus strangers' profiles, i.e., by the visiting-friends probability q_j. We assume that a user's profile is always accessible to all of his/her friends. However, a person's profile may not be available to all strangers. We assume that the probability that a stranger's profile is accessible to a user j is w_j. As our proposed analytical model and simulation results will show, visiting friends more often than stranger helps to contain a malware within a community, slowing down its propagation.



(a) Numerical results for Eq. (4), $N = 3,000$ (b) Simulation results, $N = 3,000$ nodes (c) Simulation results, $N = 10,000$ nodes

Fig. 1. User behaviors: impacts of the visiting-friends probability.

TABLE II
VARIABLE DEFINITIONS

Parameters	Definition
N	Initial number of uninfected profiles
$I_0 = 1$	Initial number of infected profiles, which is one.
q_j	Probability that user j visits a friend's profile
$1 - q_j$	Probability that user j visits a stranger's profile
r_j	Degree of node j (the number of friends user j has)
$I[i]$	Total number of infected profiles at the end of the ith event
$V[i]$	Number of uninfected profiles remaining at the end of the ith event. Thus $I[i] + V[i] = I_0 + N$
$p(i+1 \mid j)$	Probability that user j gets infected at the end of the (i+1)th event
$p(j \in V[i])$	Probability that user j is uninfected at the end of the i event, $p(j \in V[i]) = 1 - I[i]/N$
$E[i+1]$	Average number of infected profiles at the end of the $(i+1)$th event
ϕ	Probability of choosing a user for an event, $\phi = 1/N$
w_j	Probability that a stranger's profile is accessible to the user j
$I[i,j]$	Number of friends of user j that are infected at the end of the ith event

A. Analytical Model

Table II lists the definitions of the variables used in the following analysis. We compute the total number of infected profiles $I[i+1]$ at the end of the $(i+1)$th event, given N, $I_0 = 1$, q_j and r_j as *initial conditions*.

The probability that an uninfected user j gets infected at the end of the $(i+1)$th event is:

$$p(i+1|j) = p\,(j \in V[i]) \times \left(q_j \times \frac{I[i,j]}{r_j} + (1-q_j) \times w_j \times \left(\frac{I[i] - I[i,j]}{N - r_j}\right)\right) \quad (1)$$

Eq. (1) states that a user j will become infected if he/she visits an infected friend with a probability q_j or an infected stranger with a probability $1 - q_j$. If we take the sum of $p(i+1|j)$ over all users, the result is the average number of infected profiles in the OSN at the end of the $(i+1)$th event, which is denoted by $E[i+1]$.

$$\begin{aligned} E[i+1] &= \sum_{j=1}^{N} \phi \times p(i+1|j) \times 1 \\ &= \sum_{j=1}^{N} \frac{1}{N} \times p\,(j \in V[i]) \\ &\quad \times \left(q_i \times \frac{I[i,j]}{r_j} + (1-q_i) w_j \left(\frac{I[i] - I[i,j])}{N - r_j}\right)\right) \end{aligned} \quad (2)$$

The number of friends of user j that are infected at the end of the $(i+1)$th event is as follows:

$$I[i+1, j] = I[i,j] + E[i+1] \times \frac{r_j - I[i,j]}{N - I[i]} \quad (3)$$

The number of infected users in the whole OSN at the end of the $(i+1)$th event is as follows:

$$I[i+1] = I[i] + E[i] \quad (4)$$

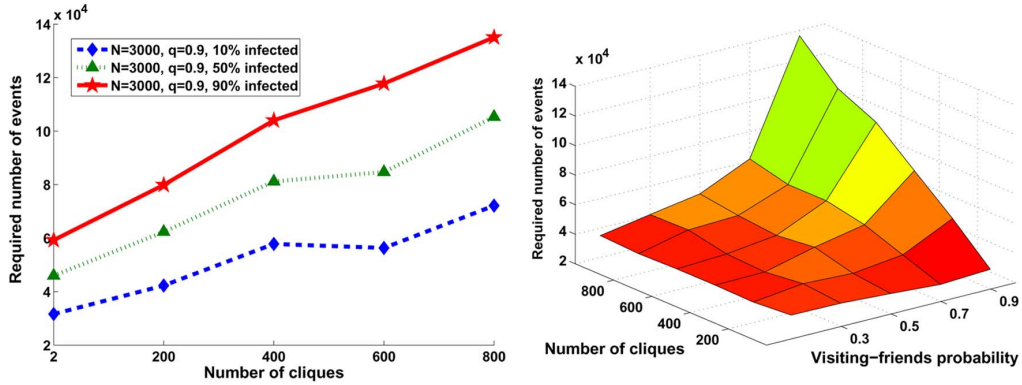
Numerical Results: Assuming that all users in the network have the same visiting-friends probability $q_j = q$ and $w_j = 1$, we plotted graphs of function $I[i+1]$ given $N = 3,000$ nodes (users), $I_0 = 1$, and three different values of $q = 0.1$, 0.5 and 0.9. The graphs in Fig. 1(a) show that as people visit their friends more often than strangers ($q = 0.9$), the worm propagation is slower. For instance, after the 4000th event, there are 2,980 infected users in the network when $q = 0.1$ versus only 350 when $q = 0.9$. The reason is that the worm circulates for a while within a group of friends (a community) before reaching out to other parts of the network.

B. Simulation Results

To validate the proposed model, we performed simulations using the two OSN graphs of sizes 3,000 and 10,000 nodes as described in Section II. We assume that all users have the same visiting-friends probability q. In each time slot, an uninfected user is chosen randomly based on a uniform distribution, who will visit one of his/her friends with probability q, or a stranger with probability of $1 - q$. We recorded the number of infected users at the end of each event given $q = 0.1$, 0.5 and 0.9 to plot the graphs shown in Fig. 1(b) and Fig. 1(c). The simulation results show that increasing the value of q slows down the propagation. The results are consistent with the proposed model presented above.

IV. CLIQUE SIZES

In a highly clustered OSN, users form small cliques (also called communities or groups) [14]. Members of a clique tend to visit each other (their friends) more often than strangers (people outside the clique). Assume a network of N users that are divided into n small groups (cliques). A clique is defined as a maximal complete subgraph of three or more nodes. Given the tiny social network in Fig. 4, two examples of cliques are $\{v_1, v_2, v_3, v_4, v_5\}$ and $\{v_5, v_6, v_7\}$.



(a) $N = 3,000$ nodes, $q = 0.9$, varying k (b) $N = 3,000$ nodes, varying both k and q

Fig. 2. Impacts of the number of cliques.

Assume that each clique i in a social network has S_i members and clique members are only connected to each other and not to other cliques. Thus, $\sum_{i=1}^{n} S_i = N$. Each member of a clique will visit members in the same clique (friends) with a probability q and visit members outside the clique (strangers) with probability $1 - q$. Let I_i denote the current number of infected users in clique S_i. If a user u from clique S_1 visits a profile in the OSN, the probability that u will get infected is as follows:

$$\frac{S_1}{N} \times \frac{S_1 - I_1}{S_1} \times q \times \frac{I_1}{S_1} + \sum_{i=2}^{n} \frac{S_1}{N} \times \frac{S_1 - I_1}{S_1} \times (1-q) \times \frac{S_i}{N} \times \frac{I_i}{S_i} \tag{5}$$

Thus the *average number of new infections* in the next visit is as follows:

$$\sum_{i=1}^{n} \left(\frac{S_i}{N} \times \frac{S_i - I_i}{S_i} \times q \times \frac{I_i}{S_i} + \sum_{j=1, j\neq i}^{n} \frac{S_i}{N} \times \frac{S_i - I_i}{S_i} \times (1-q) \times \frac{S_j}{N} \times \frac{I_j}{S_j} \right) \tag{6}$$

$$= \sum_{i=1}^{n} \left(\frac{(S_i - I_i) \times q \times I_i}{S_i \times N} + \frac{(S_i - I_i) \times (1-q)}{N^2} \sum_{j=1, j\neq i}^{n} I_j \right) \tag{7}$$

Expression (7) shows that the infection rate (propagation speed) depends on both the community size S_i and the visiting-friends probability q.

We carried out two experiments using a network of 3,000 nodes[1] to study the effect of the clique size S_i and the visiting-friends probability q on the infection rate. We divide N users into k cliques where $k \leq N/3$. This ensures that each clique has at least three members. If N is not divisible by k then some cliques will have one member more than the others. For instance, if $N = 100$ and $k = 30$, then 20 cliques have 3 members each, and the other 10 cliques have 4 members each. This allows all cliques to have approximately the same size.

In the first experiment, we measured the number of events required in order to infect 10%, 50% and 90% of the network population, respectively, assuming a visiting-friends probability $q = 0.9$. (A higher number of events required implies a slower propagation.) The results given in Fig. 2(a) show that as the clique size decreases and thus the number of cliques increases, it requires more events to infect the same number of people. In other words, increasing the number of cliques leads to slower propagation. For instance, when the number of cliques goes from 200 to 600, the number of events required in order to infect 90% of the network population increases from 7,980 to 11,780 events, or 1.5 times.

To explain the simulation results, consider the following example with two scenarios. In the first scenario, all 3000 members of the OSN form one clique. In the second scenario, the OSN is divided into 100 cliques, each having 30 members. Assume a visiting-friends probability $q = 1$. That is, every user visits only his/her friends in the same community and never a person outside his/her community. In the first scenario, if a user u is infected by an XSS worm, all other 2,999 members will eventually get infected since they all belong to the same clique and interact with each other. In the second scenario, only 29 members residing in the same clique as user u will get infected, and the rest of the OSN will not since $q = 1$. We can consider user u's community as being quarantined from the rest of the OSN. Therefore, the time (or number of events) needed to infect x percent of the population, where $x > 1$, is infinity. That is, a large number of small cliques helps slow down the worm propagation (or stops it in cases where $q = 1$). In the above experiment, we set the visiting-friends probability q to 0.9, allowing the worm to propagate from the initial infected user's clique to other cliques. However, the same explanation applies: a large number of small cliques makes the propagation slower than a few big cliques.

In the second experiment, we varied *both* the number of cliques k and the visiting-friends probability q, and measured the number of events ε_{90} required to infect 90% of the population. We observe the following trends from the results given in Fig. 2(b). First, given the same visiting-friends probability q, as the number of cliques increases, more events are required to infect 90% of the population. That is, a large number of small cliques helps slow down the propagation compared with a smaller number of big cliques. This observation is consistent with that from the first experiment discussed above. Second, given the same number of cliques k, increasing the visiting-friends probability q slows down the malware propagation

[1]We were able to simulate only the smaller network in these experiments because such an experiment required a very large amount of memory.

speed. This is consistent with the model and simulation results presented in Section III. Third, and most interestingly, the impact of the visiting-friends probability q is more pronounced when the number of cliques is high. For instance, when q goes from 0.3 to 0.9, ε_{90} increases from 36,400 to 99,800 events, or 2.75 times, given $k = 600$ cliques. When $k = 1,000$ cliques, ε_{90} increases from 40,700 to 136,000 events, or 3.4 times. This observation again emphasizes the advantage of having a large number of small communities in an OSN. In times of XSS worm attacks, if each of these communities is monitored, this will slow down the worm propagation, allowing the network administrator more time to detect and eliminate the worm. This concept is consistent with disease prevention and control practices in the field of health care.

V. Clustering Coefficients

In addition to the visiting-friends probability and clique size, the highly clustered structure of an OSN, or its clustering coefficient, also plays an important role in the propagation speed of a malware. To illustrate this point, we compare the propagation speed of a malware in a synthesized OSN with that in an equivalent random graph (ERG). Given an OSN graph, we can use an algorithm such as the one by Viger and Latapy [15] to reconnect the vertices of the original OSN graph (i.e., to generate a different set of edges) so that the resulting ERG still has the same number of nodes, number of edges, and maximum and average node degrees as the original OSN. However, since the edges are different, the ERG will have a different clustering coefficient, usually much lower than the clustering coefficient of the original OSN graph.

Given the two OSN graphs with sizes $N_1 = 3,000$ and $N_2 = 10,000$ nodes and clustering coefficients $C_1 = 0.19$ and $C_2 = 0.15$, respectively, as described in Section II, we created two ERGs of the same sizes with clustering coefficients $C_1' = 0.005$ and $C_2' = 0.004$, respectively. We recorded the number of infected users at the end of each event, assuming a visiting-friends probability $q = 0.9$ in all four networks. The results in Fig. 3(b) illustrate the number of infected users as a function of the number of events (visits). The graphs show that, although an OSN graph and its ERG share the same probability q and other parameters (e.g., number of edges, maximum and average node degrees), the infection rates are different in the two networks. The propagation is slower in the original OSN graph than in the ERG thanks to its *higher clustering coefficient*. For example, in the 3000-node networks, after 40,000 events, there are 1, 300 infected users in the OSN graph versus 2,100 infected users in the ERG. Given a high visiting-friends probability, people tend to visit their friends within a community much more often than strangers. Given a high clustering coefficient, a malware will circulate for a while in a community among friends before reaching out to other parts of the OSN, slowing down the malware propagation.

The above analytical models and simulation results show that users' tendency to visit their friends more often than strangers and the community structure help slow down the propagation of XSS worms in OSNs. The issue is how we can take advantage of these properties to make malware detection more resource-efficient than the exhaustive checking method used by Facebook [4].

In this article, "resource efficiency" is defined as follows. Suppose that the OSN is capable of scanning (monitoring) N nodes in a fixed period of time T. To scan one node, the system uses α scanning techniques to detect a malware, and each technique is executed by a request from the system. Therefore, in the time interval T, the monitoring system is capable of handling $N \times \alpha$ requests. Suppose that the processing power needed to handle these requests is equal to Ω units, where a unit can be defined as the number of machine instructions executed in time interval T.

By monitoring only a subset of strategically selected nodes, we reduce the number of nodes to be scanned from N to a fraction of N, say, N/k (at the cost of potentially more infections before the first detection). Thus, the monitoring system receives on average $(N/k) \times \alpha$ scanning requests, resulting in less processing power needed, i.e., Ω/k.

Given less processing power needed to scan less nodes, the system has the option of utilizing the available processing power to apply more rigorous and power-demanding scanning techniques, that can lead to the detection of highly sophisticated or zero-day malware.

In this article, we explore the possibility of using the selective monitoring approach instead of the exhaustive checking method. In particular, we present a study of potential selective monitoring schemes. In a selective monitoring scheme, we do not monitor every read/write post or every user in the OSN. Instead, we monitor only a subset of users and their friends' activities. We can take advantage of the characteristics of OSNs as discussed above to select this subset of users to be monitored, so that the coverage is maximized while we can minimize resource usage. The subset of users to be monitored is selected using different metrics that take into account the highly clustered structure, short average distance and node degree distribution of a social network.

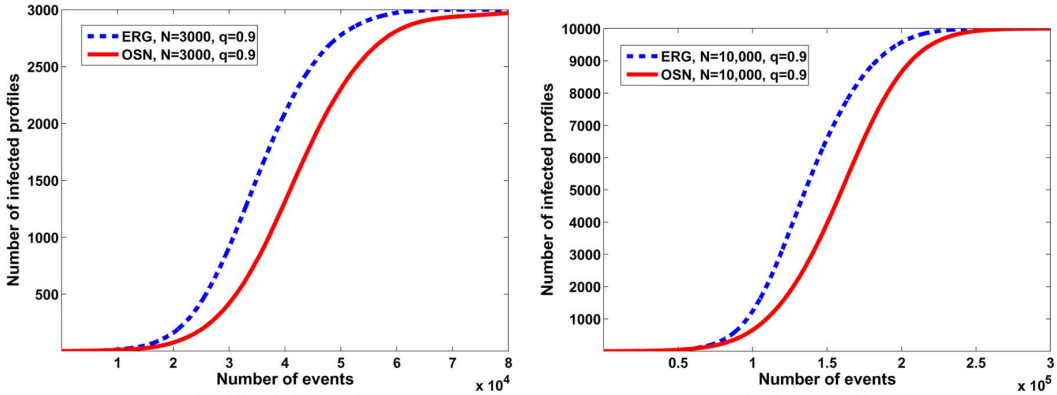
VI. A Study of Selective Monitoring Schemes

In these schemes we first select a set of important users and monitor these users' and their friends' activities and read/write posts for malware threats. We call these important users candidates to be monitored or "candidates" for short. After selecting candidates, we apply monitoring techniques such as those used in the Facebook system [4], PathCutter [6], or Spectator [7] in a distributed manner to monitor *only* the candidates' and their friends' posts.

There are two questions to be answered. How do we select candidates to be monitored? How many candidates should we deploy in an OSN? We address the first question in Section VI-A by examining five metrics for selecting monitored candidates. The answer to the second question involves a trade-off between resource consumption and the required detection time. The more candidates we deploy, the faster we can detect a malware propagating in the network.

A. Candidate Selection Metrics

We have identified five metrics that can be used to select candidates to monitor, all based in the relative importance of a



(a) $N = 3,000$ nodes (b) $N = 10,000$ nodes

Fig. 3. Impacts of clustering coefficients: OSN graphs versus ERGs.

TABLE III
DIFFERENT METRIC MEASURES BASED ON FIG. 4

Metric	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}
Degree	4	4	4	4	6	2	7	1	1	1	1	1
Closeness	0.47	0.47	0.47	0.47	0.68	0.55	0.73	0.44	0.44	0.44	0.44	0.44
betweenness	0	0	0	0	28	0	40	0	0	0	0	0
PageRank	0.09	0.09	0.09	0.09	0.14	0.06	0.23	0.04	0.04	0.04	0.04	0.04
Cross-Connectivity	11	11	11	11	12	1	1	0	0	0	0	0

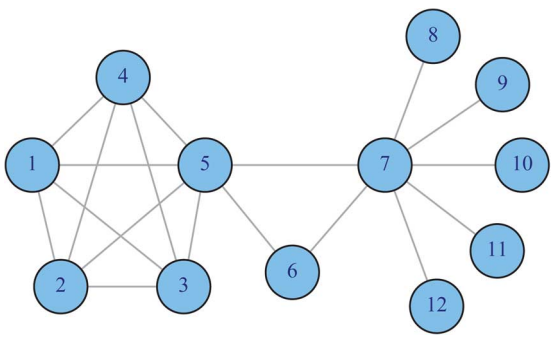


Fig. 4. Tiny social network graph.

node in the network. The five metrics are node degree, closeness [16], betweenness [16], PageRank [16] and cross-clique connectivity. The closeness, betweenness and PageRank metrics leverage upon the short average distance property of an OSN to detect malware propagation. The node degree and cross-clique connectivity metrics, on the other hand, take advantage of the highly clustered structure of an OSN for malware detection. Following are the definitions of the five metrics.

1) Node Degree: This is the simplest metric among the five. The degree $deg(v)$ of a node v is the number of edges incident on v. Given the example OSN in Fig. 4, if we selected two candidates to monitor, they would be nodes v_7 and v_5. Their degrees are higher than those of the others (see Table III): $deg(v_7) = 7$ and $deg(v_5) = 6$.

A network sanitization scheme suggested by Yan *et al.* [2] selects nodes with the highest degrees, inspects messages going in and out from these nodes, and removes embedded malicious URL, if any.

2) Closeness: To measure the closeness of a node in a graph $G(E, V)$, we first calculate the farness of that node. The farness of the node is defined as the sum of the lengths of the shortest paths from that node to the other nodes in G. The closeness is defined as the inverse of the farness. Hence, the higher the closeness of a node is, the lower its total distance to all other nodes. This measure represents how fast a message sent by node s will reach all other nodes, assuming node-to-node delay is the same on all links. Let $d(u, v)$ denote the length of (number of hops on) the shortest path between u and v, the closeness $D(v)$ of node v is defined as:

$$D(v) = \frac{1}{\sum_{u \in V} l(u, v)} \tag{8}$$

In the example OSN in Fig. 4, the top two candidates to be selected based on the closeness metric are v_7 and v_5. Their closeness values are $D(v_7) = 0.73$ and $D(v_5) = 0.68$, higher than those of the other nodes (see Table III).

As an attack strategy, it would be wise to infect nodes with high closeness values first, since they are the closest to the other nodes. This would allow a malware/virus to propagate fast throughout the whole network. By the same reasoning, Tang *et al.* [17] also select nodes with high closeness values as starting points to distribute "disinfecting" patches in a mobile network in order to contain an active malware.

3) Betweenness: The betweenness $B(v)$ of a node v is defined as the number of shortest paths from all possible pairs in a graph that traverse node v.

$$B(v) = \sum_{s \neq v \neq t, s, t, v \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}, \tag{9}$$

where σ_{st} denotes the total number of shortest paths from a node s to a node t, and $\sigma_{st}(v)$ represents the number of paths that traverse node v. The betweenness C_B of node v determines the influence of v on the information flow passing through the node. The betweenness values of the nodes in the example OSN are given in Table III, which shows that v_7 and v_5 are the top two candidates to monitor. Their betweenness values are the highest with $D(v_7) = 40$ and $D(v_5) = 28$.

Tubi *et al.* [10] studied a similar metric named "group betweenness" which measures the betweenness of a group of nodes instead of an individual node. Nodes with high "group betweenness" values are monitored in order to slow down malware propagation in an *e-mail network*.

4) PageRank: PageRank is an algorithm that assigns a numerical weighting to each entity of a collection of entities with reciprocal references. PageRank is used by the Google Internet search engine to rank web pages on the Internet [18]; in this case, entities to be weighted are web pages. If a website A links to a website B and B links to A, they are reciprocally linked (reciprocally referenced). In this context, a PageRank value of a website A represents the likelihood that Internet users who start on a random page and then follow random links (on that page or from the entire web) will arrive at website A.

If we apply this concept to an OSN, then entities are users of the OSN, and reciprocal references are friendships between users. Heideman *et al.* [19] use the PageRank metric to identify key users in an OSN. In this context, a PageRank value of a user v can be used to indicate the likelihood that a malware that starts randomly somewhere in the OSN and propagates in the network will reach and infect v. Therefore, we should monitor users with high PageRank values in order to catch a malware in its early stage of propagation.

PageRank values of nodes (web pages) in a network can be computed either iteratively or algebraically [20]. These two methods are described in Appendix A. Table III lists the PageRank values of the nodes in the example OSN in Fig. 4, calculated using the iterative method. Nodes v_7 and v_5 have the highest PageRank values, and thus the highest probability of being infected by a malware starting at a random node in the network. Thus they should be candidates for monitoring.

5) Cross-Clique Connectivity: We propose a new metric that measures the connectivity of a node to different communities or cliques. The cross-clique connectivity $X(v)$ of a node v is the number of cliques to which v belongs. A node with a high $X(v)$ value is called a *highly cross-connected node*. In every day life, a well-cross-connected person would travel among many communities, and thus be a potential disease carrier from one community to another in case of a disease outbreak or epidemic. Therefore, this person should be monitored in such a case. We apply the same concept to monitoring users in an online social network.

The algorithm for computing the cross-clique connectivity of a node is given in Appendix B. The example social network in Fig. 4 has a total of 17 cliques, which are listed in Appendix C. In this example network, node v_5 has the highest cross-clique connectivity, $X(v_5) = 12$. The cross-clique connectivity values of the other nodes are listed in Table III. If we wish to select one more highly cross-connected node to monitor in addition to v_5, any of the following nodes can be chosen: $\{v_1, v_2, v_3, v_4\}$ since they have the same cross-clique connectivity value of 11.

In the above examples, nodes v_5 and v_7 are almost always selected as the best nodes to monitor. However, in real large networks, the set of candidates given by a metric may not be the same as the set given by another. For instance, the node with the highest degree in an OSN may not be the node with the highest closeness or betweenness value in that network.

Given several metrics defined in the literature for selecting candidates to be monitored, which one should we choose to apply to malware detection in a social network? We performed simulations to answer this question.

B. Simulation Settings

We carried out simulations to study the effectiveness of the above five metrics with respect to malware detection. We used three OSN graphs of 10,00, 20,000 and 100,000 nodes whose parameters are listed in Table I. For each metric, we selected the top 1, 7, 15 and 20 candidates, respectively, and set up the system to monitor their friends' activities and posts.

We define an *event* in the simulation to be the action of visiting or accessing a user's profile by some other user. We assume that events in an OSN happen consecutively one after another. (Two different users may click on the same profile at the same time. Their access requests, however, will be queued at a server consecutively, waiting to be processed. The two events are thus considered to happen one after the other.)

The simulation software is implemented using MATLAB. The simulation is of discrete-event type, consisting of discrete virtual time slots. A time slot is equivalent to an *event* defined above. At the start of each run, we choose a node randomly using a uniform distribution and infect it with the malware. In each time slot, a user (node) j is chosen randomly with a probability of $1/N$ and the user will visit a friend's profile with a probability $q = 0.9$ and a nonfriend user's profile with probability $1 - q = 0.1$. Two users are friends if and only if their corresponding vertices are adjacent in the OSN graph. User j will become infected if she visits an infected friend with a probability q or an infected stranger with a probability $1 - q$.

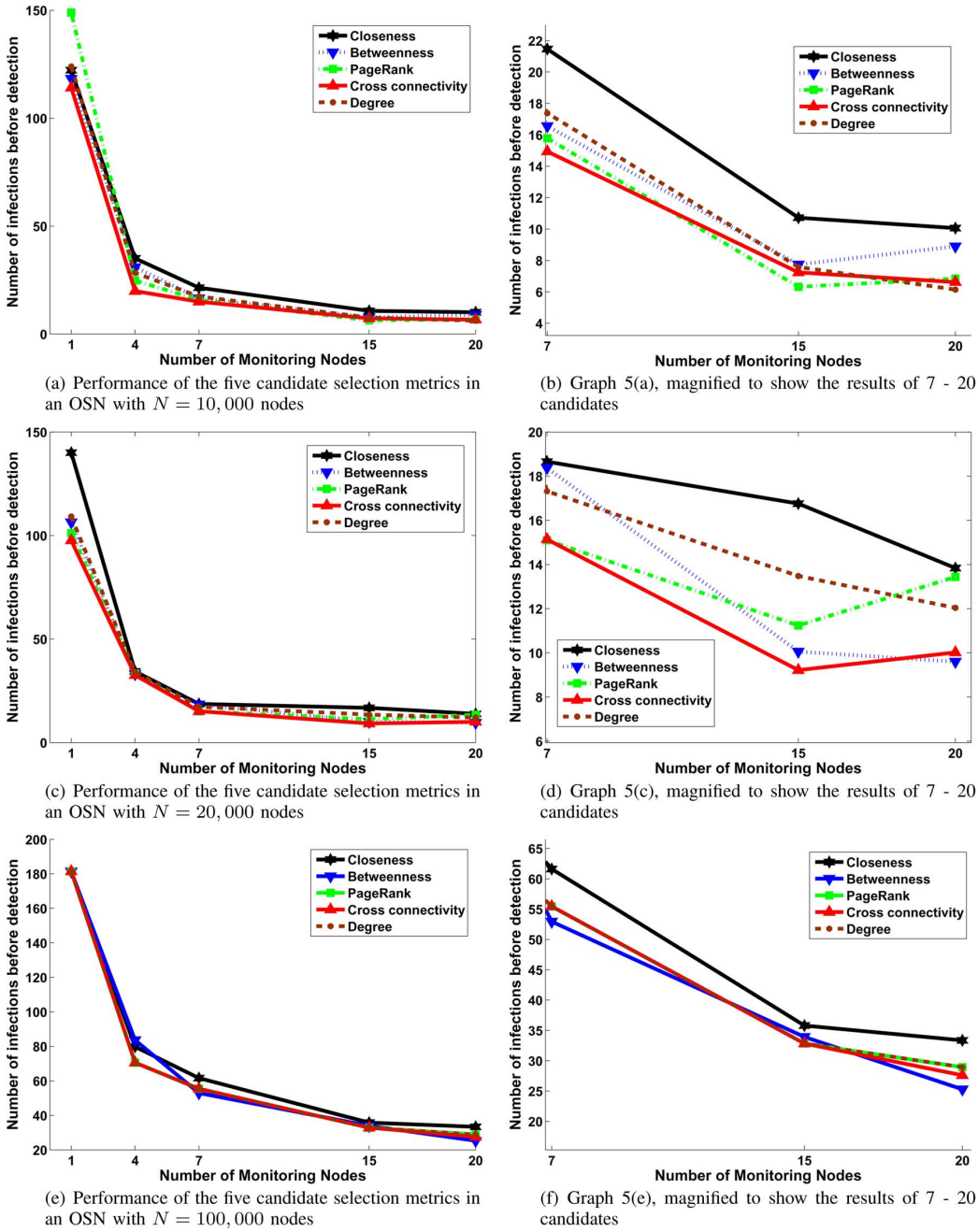
The monitoring action is simulated as follows. In each time slot, the simulation code at each candidate's node checks the messages read/posted by the candidate's friends. If one of these messages contain the malicious code (which is identified by a unique signature), the code will detect its presence and raise an alarm. The malware is considered detected and the simulation is ceased. We then count the number of nodes that were infected (i.e., read/posted the malicious code) at this point and record that number.

Each data point in the result graphs is the average of 100 runs, each with a different random seed.

C. Simulation Results

The numbers of infections before the first detection given by the five metrics are shown in Fig. 5(a), 5(c) and 5(e) for the 10,000-node, 20,000-node and 100,000-node networks, respectively. We also magnify those graphs for x-axis values from 7 to 20 candidates as shown in Fig. 5(b), 5(d) and 5(f).

The results in Fig. 5(a), 5(c) and 5(e) show that the five metrics have similar effectiveness in terms of malware detection time. The reason is that the set of candidates computed from one metric can overlap partially or completely with the set computed by another metric, depending on the network topology. For instance, in the 100,000-node network, 95% of the top 20 candidates are the same in all five cases (metrics), leading to their similar performance with respect to detection time. We note, however, that the closeness metric is the worst performer among the



(a) Performance of the five candidate selection metrics in an OSN with $N = 10,000$ nodes

(b) Graph 5(a), magnified to show the results of 7 - 20 candidates

(c) Performance of the five candidate selection metrics in an OSN with $N = 20,000$ nodes

(d) Graph 5(c), magnified to show the results of 7 - 20 candidates

(e) Performance of the five candidate selection metrics in an OSN with $N = 100,000$ nodes

(f) Graph 5(e), magnified to show the results of 7 - 20 candidates

Fig. 5. Performance of different selective monitoring metrics.

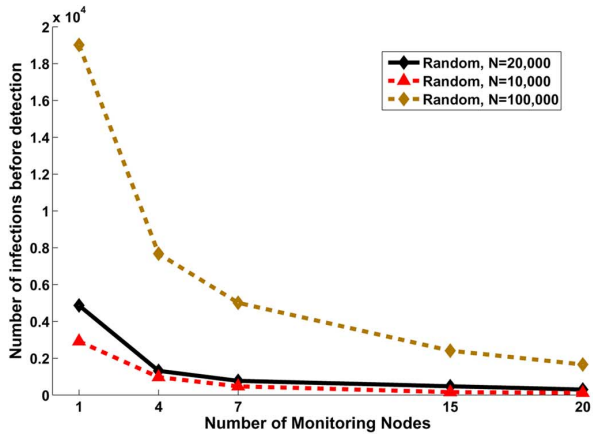
five, and the cross-clique connectivity metric is always among the best performers.

Given that several metrics offer similar performance in terms of malware detection time, the decision as to which metric we should use to select candidates can be made using other factors, such as the complexity of computing the set of candidates to be monitored. Table IV in Appendix D shows the complexities of the algorithms used for selecting candidates based on the five metrics. The information suggests that the node degree and PageRank metrics offer the best of both worlds, detection time and computation overhead.

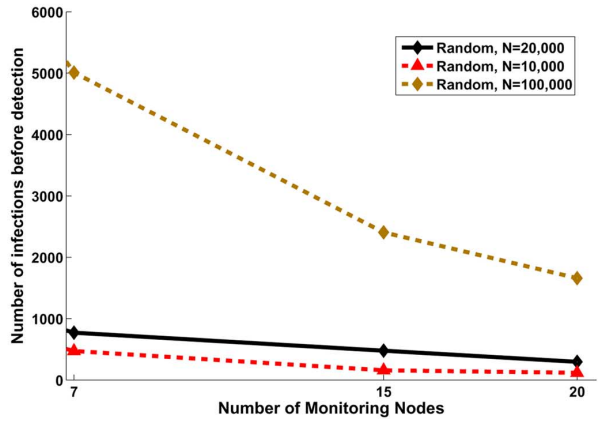
The graphs also show a trade-off between resource consumption and detection time: the more nodes are monitored, the earlier a malware can be detected (i.e., the less nodes are infected before the malware is detected). For example, in the 100,000 node case in Fig. 5(e), given 7 candidates to monitor, the number of users infected before the first detection is approximately 55 users. Given 20 candidates to monitor, the number of infections before the first detection is reduced to 34 users.

TABLE IV
COMPLEXITY OF DIFFERENT SCHEMES

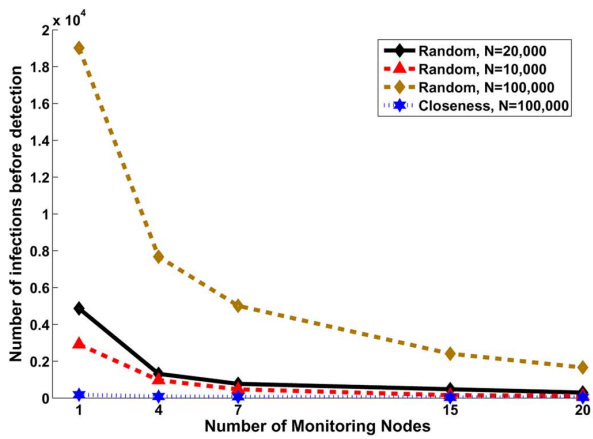
Metric	Runtime
Degree	$\Theta(\|V\|^2)$
Closeness	$O((k+\|V\|^{\frac{2}{3}} \times \log^{\frac{1}{3}}\|V\|)(\|V\|\log\|V\|+\|E\|)$
Betweenness	$O(\|V\| \times \|E\|)$
PageRank	$O(\|V\|^2)$
Cross-clique Connectivity	$O(\Delta^4 \times \mu)$



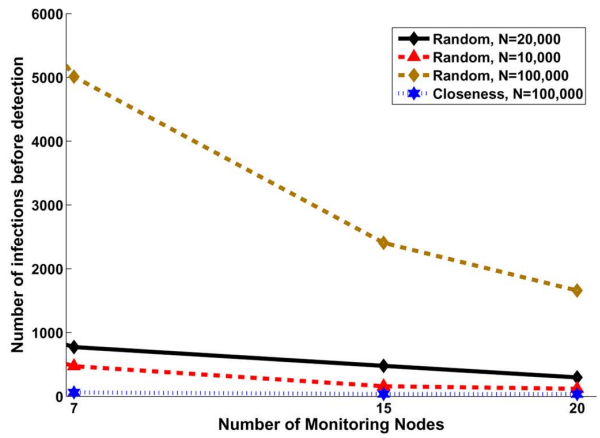
(a) Candidates randomly selected, $N = 10,000, 20,000$ and $100,000$ nodes



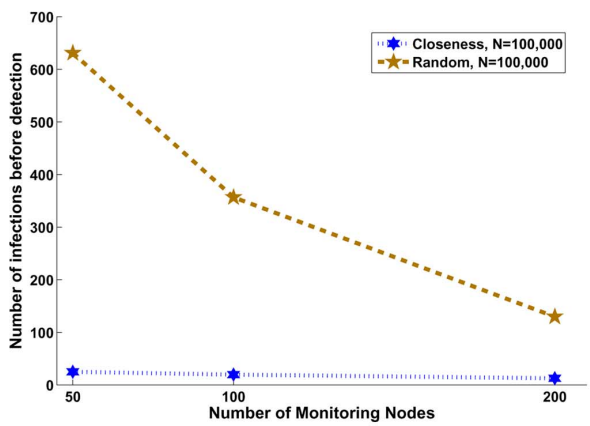
(b) Graph 6(a), magnified to show the results of 7 - 20 candidates



(c) Graph 6(a), including the closeness metric from Graph 5(e)



(d) Graph 6(b), including the closeness metric from Graph 5(f)



(e) Random vs. Closeness candidate selection for 50, 100 and 150 candidates, $N = 100,000$

Fig. 6. Performance of the random selection scheme.

To demonstrate the effectiveness of the selective monitoring approach, we repeated the above experiments, but selected the candidates to monitor *randomly*. The results of this set of experiments are illustrated by the graph in Fig. 6(a), with the magnified curves shown in Fig. 6(b). The results indicate that the random selection approach takes longer to detect a malware than any of the above five metrics. For instance, in the 100,000-node network with 15 candidates to monitor, the average number of infections before the first detection is 32.84 users for the cross-clique connectivity metric, and 2407 users for the random selection scheme, a staggering difference of more than 75 times.

To further illustrate the advantage of the selective monitoring method versus random candidate selection, we add the curve of the closeness metric in Fig. 5(e) to Fig. 6(a), resulting in Fig. 6(c), with the magnified curves shown in Fig. 6(d). Figs. 6(c) and 6(d) show that even the worst performer of the selective monitoring method—the closeness metric—still outperforms the random candidate selection approach. For

instance, in the 100,000-node network with 7 candidates to monitor, the average number of infections before first detection is 62 users for the closeness metric, and 5010 users for the random selection scheme, a difference of about 80 times.

To further compare the random selection approach with the selection scheme based on the closeness metric, we increase the number of candidates to monitor to 50, 100 and 200 nodes. The result in Fig. 6(e) demonstrates that the selective monitoring method outperforms the random selection scheme in all cases. For example, given 100 candidates to monitor, the average number of infections before the first detection is 20 users for the closeness metric, and 356 for the random selection scheme, a difference of approximately 18 times.

In summary, most metrics defined in the literature have similar performance in terms of detection time. However, the closeness metric in general is the worst performer among the five, and the cross-clique connectivity metric is always among the best performers. All the five metrics outperform random selections of monitored candidates by a large margin.

VII. Related Work

Among the first works on malware propagation in online social networks are [1], [21]. Faghani and Saidi [21] model XSS worm propagation using the susceptible-infected (SI) model [22] and the visiting-friends probability. Their proposed model shows that the infection rate is inversely proportional to the visiting-friends probability. They also present simulation results of malware propagation speed as functions of the visiting-friends probability and initial number of infections in [1]. Our propagation model in this paper is not based on the SI model and our simulation results are far more extensive, taking into account the highly clustered structure of social network and clique sizes.

Faghani *et al.* [23] use simulation to show that users' probability of executing a malware can speed up the propagation speed of Trojan worms *exponentially*.

Several other researchers also study malware propagation in OSNs via simulations [2], [3], [10]. However, they did not use graphs with the characteristics of OSNs (i.e., high clustering coefficient, and low average shortest path distance) [3], [10], or limited their simulations to only a single network [2].

Nguyen *et al.* [24] study the PageRank metric for the purpose of containing misinformation in an OSN.

Yan *et al.* [2] describe three approaches for node monitoring in order to detect malware in OSNs using (1) node degree metric, (2) user activities and (3) network partition into small islands. In the first approach, nodes with the highest degrees are chosen to be monitored, as discussed in Section VI-A1. In the second approach, the most active nodes are selected for monitoring. Examples of the most active nodes are major broadcasting companies such as CNN and BBC, which post news updates frequently throughout the day via OSNs such as Facebook and Twitter. In the third approach, an OSN is partitioned into small islands, and every message exchanged between islands is inspected for potential viruses/malware. The goal is to contain a computer worm within a region of the network and stop it from propagating further. This approach, however, is applicable only to active worms that propagate themselves through a victim's friend list. Passive worms such as XSS worms, on the other hand, stay dormant in an infected profile, waiting for a user to click on that profile, visit it and become infected. Therefore, containing XSS worms is not pertinent as they do not actively propagate themselves. A more relevant method is to detect their presence via monitoring and then eliminate them. Unlike our work in this paper, Yan *et al.*'s paper does not provide quantitative results to illustrate the effectiveness of their suggested selective monitoring schemes (except for the third approach for which the authors gave an example that required 78% of all edges (friendships) to be inspected in order to contain the propagation within 10% of the population).

Xu *et al.* [3] propose a correlation-based scheme to mitigate the effects of active worm propagation in OSNs. They assign "decoy friends" to a subset of users, and the "decoy friends" will monitor network activities. The main disadvantage of this scheme is the difficulty of getting users' consent to add "decoy friends" to their friend networks as this may infringe upon their privacy. Xu *et al.*'s scheme defends against active worms in OSNs such Koobface and is not designed for detection of passive worms such as XSS worms.

Cao *et al.* presented PathCutter [6], an XSS detection tool that can detect traditional XSS, DOM-based XSS and content sniffing XSS vulnerabilities. They achieve their goal by two integral mechanisms: view separation and request authentication. They divide the web application into different views, then PathCutter isolates different views on the browser side. The solution also provides a per-URL session token and referrer-based view validation to protect against other XSS-related types of attacks. PathCutter can be implemented using server code modification or as an standalone proxy server. The main disadvantage of the PathCutter solution is the rendering latency introduced by PathCutter at the client side. For example, in a post with 45 comments, the system will respond with 30% higher latency compared with the case where PathCutter is not implemented.

Livshits *et al.* [7] designed a detection and containment method called Spectator [7]. Spectator uses a tainting and tagging approach to detect the spread of JavaScript worms which is implemented as a proxy. Whenever a new tag is inserted into the Spectator system, a node is added to a tracking graph called "propagation graph". When the diameter of the propagation graph exceeds a user-defined threshold d, the system raises an alarm. Choosing an appropriate threshold value is not a simple task: a low threshold will cause many false positive alarms in the system, while a high threshold will detect a malware only after a large number of users have been infected.

Sun *et al.* [27] proposed a client-side solution to detect XSS worms using a Firefox plug-in. As part of their approach, they use string comparison to detect the worm propagation which is vulnerable to polymorphic attacks.

VIII. Conclusion

We present analytical models and simulation results that characterize the propagation of XSS worms in OSNs. We show that the propagation speed of an XSS worm in an OSN depends on three major factors. First, if users visit their friends more often than strangers, this will help slow down the propagation.

Second, a large number of cliques also contributes to decreasing the speed of propagation. Third, the highly clustered structure of an OSN helps contain an XSS worm within a community for some time before it reaches other communities, slowing down the propagation.

The above results show that it is feasible to detect a malware in its early stage of propagation by monitoring only a subset of users of an OSN and taking advantage of the characteristics of OSNs. We present a study of five metrics used to select candidates for monitoring: node degree, closeness, betweenness, PageRank and cross-clique connectivity. Our simulation results show that the metrics perform similarly in terms of detection time, with the cross-clique connectivity being among the top performers. There are metrics that offer a good trade-off between detection time and computation overhead such as node degree and PageRank. All five metrics outperform random placements of monitored candidates by a large margin.

In our future work, we will extend the analyses and simulations to OSNs represented by directed graphs such as Twitter. We will also extend the analysis of the clique size to cover cross-clique scenarios. In addition to the network topology and strategic positions of monitored nodes in the network, user activities also play an important role in the propagation of malware. In the future, we will study the impacts of user activities in malware propagation using real-world data.

APPENDIX

A. PageRank Algorithms

1) The Iterative Method: In the first round at $t = 0$, an initial probability distribution is assumed for every node as $R(v_i, 0) = 1/N$, where N is the total number of nodes in the graph. In the $(t+1)$th iteration, the PageRank value of each node is computed using the following equation:

$$R(v_i, t+1) = \frac{1-d}{N} + d \sum_{v_j \in M(p_i)} \frac{R(v_j, t)}{L(v_j)} \qquad (10)$$

where v_i is the ith vertex, $M(v_i)$ is the set of nodes adjacent to v_i, $L(v_j)$ is the number of edges incident on node j, and d is the damping factor where $0 < d < 1$. The algorithm will converge to a certain value for each vertex which is the corresponding PageRank value of that vertex [18].

2) The Algebraic Method: Given vector $\Re(t+1)$ defined as follow:

$$\Re_{t+1} = \begin{pmatrix} R(v_1, t+1) \\ R(v_2, t+1) \\ \vdots \\ R(v_n, t+1) \end{pmatrix}$$

we have

$$\Re(t+1) = d \times \Psi \times \Re(t) + \frac{1-d}{N} \times \mathbf{1} \qquad (11)$$

Matrix Ψ equals to $(K^{-1}A)^T$, where matrix A denotes the adjacency matrix of the graph and K is the diagonal matrix with the number of edges incident on a node in the diagonal. Matrix $\mathbf{1}$ is a column vector of length N that contains only number one. The algorithm stops when, for a small value of ϵ, $\|\Re(t+1) - \Re(t)\| < \epsilon$.

B. Algorithm for Finding Highly Cross-Connected Nodes

Given a graph $G = (V, E)$ representing an OSN, each vertex represents a user and an edge represent the existence of a relationship (friendship) between the two respective users. We first find small cliques within an OSN. Finding cliques in a graph is a NP-complete problem [25]. Thus a few heuristics [26] have been proposed to solve the problem. After obtaining the cliques, we search for a set of well-cross-connected members, those who belong to several cliques. We apply a heuristic such as the one by Tsukiyama *et al.* [26] to find the cliques of G. The results is a set of k cliques $C = \{c_1, c_2, \ldots, c_k\}$, where $c_i \subset V$ and $1 \leq i \leq k$. A vertex may be present in several cliques if the corresponding user belong to several communities. We assign a counter u_j to each user $j \in V$. We then examine the cliques one by one. If a user j belongs to a clique, we increment the counter u_j by one. After all the cliques have been examined, counter u_j contains the number of cliques to which user j belongs to. We then sort the counters in the descending order. The algorithm for finding list L_1 of well-cross-connected members is summarized by Algorithm 1.

Algorithm 1 Finding well-cross-connected nodes

1: **Input**: a set of cliques $C = \{c_1, c_2, \ldots, c_k\}$; a set of counters $Q = \{u_1, u_2, \ldots, u_N\}$

2: **Output**: Ordered set of well-cross-connected nodes stored in $List$;

3: **for** $i = 1 \rightarrow k$ **do**

4: **for each** user i in c_i **do**

5: $u_j = u_j + 1$

6: **end for**

7: **end for**

8: $List \leftarrow$ sort Q in nonincreasing order

9: **return** $List$

C. List of Cliques

Following are the 17 cliques in the example social network graph in Fig. 4: $\{v_1, v_2, v_3\}$, $\{v_1, v_2, v_4\}$, $\{v_1, v_2, v_5\}$, $\{v_1, v_3, v_4\}$, $\{v_1, v_3, v_5\}$, $\{v_1, v_4, v_5\}$, $\{v_2, v_3, v_4\}$, $\{v_2, v_3, v_5\}$, $\{v_2, v_4, v_5\}$, $\{v_3, v_4, v_5\}$, $\{v_5, v_6, v_7\}$, $\{v_1, v_2, v_3, v_4\}$, $\{v_1, v_2, v_3, v_5\}$, $\{v_1, v_2, v_4, v_5\}$, $\{v_1, v_3, v_4, v_5\}$, $\{v_2, v_3, v_4, v_5\}$, and $\{v_1, v_2, v_3, v_4, v_5\}$.

D. Complexity of the Algorithms

The complexity of the algorithms for computing the five metrics discussed in Section VI is given in Table IV, where $|V|$, $|E|$, Δ, μ and k denote the number of vertices, number of edges, maximum degree of the graph, number of maximal independent sets of the graph and the number of top highest in closeness centrality respectively.

TABLE V
RUNNING TIME OF THE ALGORITHMS IN SECONDS

Metric	3,000	10,000	20,000	100,000
Degree	0.005 s	0.006 s	0.006 s	0.01 s
Closeness	0.45 s	5 s	95 s	1,063 s
Betweenness	0.76 s	9 s	42 s	3,094 s
PageRank	0.022 s	6 s	12 s	78 s
Cross-clique Connectivity	2.3 s	63 s	129 s	2,291 s

We ran experiments to measure the actual running time of each algorithm using the four OSNs whose characteristics are listed in Table I. We used a 64-bit computer with an Intel(R) Core i7-2700K 3.5 GHz processor running Windows 7 and 16 gigabytes of RAM. Table V summarizes the results. Each number in the table is the average from 10 runs using different random seeds.

It is obvious that the larger the graph, the longer the running time. These results show that the node degree and PageRank metrics offer a good trade-off between computation overheads and detection time (see also Fig. 5). Note, however, that in a real OSN the algorithms are not executed in real-time, but only periodically after several changes have been made to the network (e.g., users join/leave the OSN, new friendships (edges) added to the graph).

ACKNOWLEDGMENT

The authors would like to thank Prof. A. Matrawy and Prof. C.-H. Lung from Carleton University (Ottawa, Canada) and Prof. H. Saidi from Isfahan University of Technology (Iran). They would also like to thank the reviewers for their thorough reviews and helpful suggestions.

REFERENCES

[1] M. R. Faghani and H. Saidi, "Social networks XSS worms," in *Proc. Comput. Sci. Eng. Int. Conf.*, Vancouver, BC, Canada, 2009, pp. 1137–1141.
[2] G. Yan *et al.*, "Malware propagation in online social networks: Nature, dynamics, and defense implications," in *Proc. 6th ACM Inform., Comput. Comm. Security Symp.*, Hong Kong, China, 2011, pp. 196–206.
[3] W. Xu *et al.*, "Toward worm detection in online social networks," in *Proc. 26th Annu. Comput. Security Applicat. Conf.*, Austin, TX, USA, 2010, pp. 11–20.
[4] T. Stein *et al.*, "Facebook immune system," in *Proc. 4th Social Network Syst. Workshop*, Salzburg, Austria, 2011, pp. 8:1–8:8.
[5] N. P. Nguyen *et al.*, "A novel method for worm containment on dynamic social networks," in *Proc. Military Comm. Conf.*, San Jose, CA, USA, 2010, pp. 2180–2185.
[6] Y. Cao *et al.*, "PathCutter: Severing the self-propagation path of XSS JavaScript worms in social web networks," in *Proc. 19th Network Distributed Syst. Security Symp.*, San Diego, CA, USA, 2012.
[7] B. Livshits and W. Cui, "Spectator: Detection and containment of JavaScript worms," in *Proc. USENIX Annu. Tech. Conf.*, Boston, MA, USA, 2008, pp. 335–348.
[8] X. Wu and Z. Liu, "How community structure influences epidemic spread in social networks," *Physica A, Statist. Mech. Applicat.*, vol. 387, no. 2–3, pp. 623–630, 2008.
[9] S. Boccaletti *et al.*, "Complex networks: Structure and dynamics," *Phys. Rep.*, vol. 424, no. 4–5, pp. 175–308, 2006.
[10] M. Tubi *et al.*, "Deployment of DNIDS in social networks," in *Proc. Intell. Security Informatics*, New Brunswick, NJ, USA, 2007, pp. 59–65.
[11] P. Holme and J. Beom, "Growing scale-free networks with tunable clustering," *Phys. Rev. E*, vol. 65, no. 2, p. 026107, 2001.
[12] A. Yong-Yeol *et al.*, "Analysis of topological characteristics of huge online social networking services," in *Proc. 16th World Wide Web Int. Conf.*, Banff, AB, Canada, 2007, pp. 835–844.
[13] J. Davidsen *et al.*, "Emergence of a small world from local interactions: Modeling acquaintance networks," *Phys. Rev. Lett.*, vol. 88, no. 12, p. 128701, 2002.
[14] A. Mislove *et al.*, "Measurement and analysis of online social networks," in *Proc. 7th ACM USENIX Internet Measurement Conf.*, San Diego, CA, USA, 2007, pp. 29–42.
[15] F. Viger and F. Latapy, "Efficient and simple generation of random simple connected graphs with prescribed degree sequence," in *Proc. 11th Computing Combinatorics Int. Conf.*, Kunming, China, 2005, pp. 440–449.
[16] M. J. Newman, "Measurement and metrics," in *Networks: An Introduction*. London, U.K.: Oxford Univ. Press, 2010, ch. 7, pp. 168–198, sec. II.
[17] J. Tang *et al.*, "Exploiting temporal complex network metrics in mobile malware containment," in *Proc. World of Wireless Int. Symp.*, Lucca, Italy, 2011, pp. 1–9.
[18] L. Page *et al.*, The PageRank Citation Ranking: Bringing Order to the Web Stanford Univ., Stanford, CA, USA, 1998, Tech. Rep..
[19] J. Heidemann *et al.*, "Identifying key users in online social networks: A PageRank based approach," in *Proc. 31st Inform. Syst. Int. Conf.*, St. Louis, MO, USA, 2010, p. 79.
[20] A. Arasu *et al.*, "PageRank computation and the structure of the Web: Experiments and algorithms," in *Proc. 11th World Wide Web Int. Conf.*, Honolulu, HI, USA, 2002.
[21] M. R. Faghani and H. Saidi, "Malware propagation in online social networks," in *Proc. 4th Malicious Unwanted Programs Int. Conf.*, Montreal, QC, Canada, 2009, pp. 8–14.
[22] W. Kermack and McKendrick, "A contribution to the mathematical theory of epidemics," *Proc. Royal Soc. London Series A*, no. 115, pp. 700–721, 1927.
[23] M. R. Faghani *et al.*, "A study of trojan propagation in online social networks," in *Proc. 5th IEEE IFIP New Technology Security Int. Conf.*, Istanbul, Turkey, 2012, pp. 1–5.
[24] N. P. Nguyen *et al.*, "Containment of misinformation spread in online social networks," in *Proc. 3rd Ann. ACM Web Sci. Conf.*, Evanston, IL, USA, 2012, pp. 213–222.
[25] P. M. Pardalos and J. Xue, "The maximum clique problem," *J. Global Optimization*, vol. 4, pp. 301–328, 1994.
[26] S. Tsukiyama *et al.*, "A new algorithm for generating all the maximal independent sets," *SIAM J. Computing*, vol. 6, pp. 505–517, 1977.
[27] F. Sun *et al.*, "Client-side detection of XSS worms by monitoring payload propagation," in *Proc. 14th Eur. Research Comput. Security Conf.*, Saint-Malo, France, 2009, pp. 539–554.

Mohammad Reza Faghani (S'05) received the B.Sc. degree in communication engineering and the M.Sc. degree in communication network engineering from Isfahan University of Technology, Isfahan, Iran, in 2006 and 2009, respectively. He is currently working toward the Ph.D. degree at York University. His research interests include computer network security, malware analysis, and online social networks.

Uyen Trang Nguyen (M'04) received the Bachelor of Computer Science and Master of Computer Science degrees in 1993 and 1997, respectively, from Concordia University, Montreal, Canada. She completed the Ph.D. degree at the University of Toronto, Canada, in 2003. From 1995 to 1997, she was a software engineer at Nortel Networks, Montreal, Canada. She joined the Department of Computer Science and Engineering at York University, Toronto, Canada, in 2002, and is currently an Associate Professor. Her research interests are in the areas of mobile computing, wireless networking, multimedia applications, and information security.