Malware Propagation in Online Social Networks: Nature, Dynamics, and Defense Implications

Guanhua Yan Guanling Chen[‡] Stephan Eidenbenz Nan Li[‡]

Information Sciences (CCS-3)[*]
Los Alamos National Laboratory

[‡]Department of of Computer Science
University of Massachusetts Lowell

ABSTRACT

Online social networks, which have been expanding at a blistering speed recently, have emerged as a popular communication infrastructure for Internet users. Meanwhile, malware that specifically target these online social networks are also on the rise. In this work, we aim to investigate the characteristics of malware propagation in online social networks. Our study is based on a dataset collected from a real-world location-based online social network, which includes not only the social graph formed by its users but also the users' activity events. We analyze the social structure and user activity patterns of this network, and confirm that it is a typical online social network, suggesting that conclusions drawn from this specific network can be translated to other online social networks. We use extensive trace-driven simulation to study the impact of initial infection, user click probability, social structure, and activity patterns on malware propagation in online social networks. We also investigate the performance of a few user-oriented and server-oriented defense schemes against malware spreading in online social networks and identify key factors that affect their effectiveness. We believe that this comprehensive study has deepened our understanding of the nature of online social network malware and also shed light on how to defend against them effectively.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Security and protection; K.6.5 [**Management of Computing and Information Systems**]: Security and protection—*invasive software (e.g., viruses, worms, Trojan horses)*

General Terms

Security

Keywords

Online social networks, malware propagation, defense

[*]Los Alamos National Laboratory Publication No. LA-UR-09-08100

1. INTRODUCTION

In the past few years, online social networks have emerged as a popular communication infrastructure for Internet users. According to a recent report, the top two online social networks have attracted more than two billion monthly visits in January 2009 [30]. Currently there are 350 million active Facebook users and about 50% of them log onto Facebook in any given day [10]. According to the Nielson report, Internet users spent 17 percent of their time on visiting social networks and blogging sites in August 2009 [25]. Given the tremendous popularity of online social networks these days, a natural question is: what if a malware manages to propagate in these networks?

Such kinds of malware, unfortunately, are not fictitious. The trust relationships established among online social network users can easily be exploited to spread malware. An online social network worm, which was detected as MySpace.A in 2005, launched cross-scripting attacks against MySpace and allowed an attacker to add millions of new contacts; MySpace later also suffered several other malware attacks that exploited a Windows Metafile vulnerability and a feature of Apple's QuickTime player [19]. The Koobface worm, which first surfaced in 2008, spread itself on MySpace and Facebook by inserting comments with fake links pointing to malicious websites [17]. A victim's machine, once infected by this worm, turned into a zombie machine on a botnet. The Orkut social network site, which is managed by Google, also suffered malware attacks in the past, including MW.Orc, W32/KutWormer and W32/Scrapkut [26]. In April 2009, the Twitter social network also suffered a malware attack launched by a teenager [21]. The recent Clickjacking worm has hit hundreds of thousands of user accounts on Facebook in May 2010 [7], suggesting that malware propagation in online social networks has posed an urgent challenge for us.

Although malware attacks targeting online social networks have existed for a few years, our knowledge of the characteristics of such attacks is still lacking. Existing work in this field commonly centers on analyzing the impact of social structures such as scale-free and small-world networks on malware propagation [38, 34]. The spreading speed of malware in online social networks, however, is not only constrained by the friendship graphs formed by the users, but also affected by other factors such as online users' activity patterns and initial infection locations. Against this backdrop, we aim to unravel the effects of different aspects of online social networks on malware propagation in this work.

Our study is based on BrightKite [3], a location-based online social network launched in 2007. The open mechanism of this social network renders it possible to obtain not only the social graph formed by user friendships but also the activities by each user. This differs from many publicly available online social network datasets, which, although they may have more users, do not have detailed information about users' activities. The BrightKite dataset thus provides us a unique perspective to study malware propagation in realistic online social networks. Using this dataset, we conduct extensive trace-driven simulation of malware propagation. We further generate synthetic online social networks by using hypothetical social structures and activity models; comparing how malware spreads under different scenarios, we identify key factors that impose significant impact on malware propagation in online social networks.

Our contributions in this work are summarized as follows: **(1)** To ensure that our conclusion drawn from a specific online social network can be translated to other social networks as well, we verify that the BrightKite network is a typical social network. We also analyze the activity events generated by BrightKite users and find that the numbers of activity events generated by BrightKite users can be well characterized by a stretched exponential distribution and a large number of users used BrightKite for only a short period of time. Moreover, it is shown that the number of activity events generated by each BrightKite user is positively correlated with the number of friends that it has, although only in a weak sense. **(2)** We perform extensive trace-driven simulation to understand the effects of initial infection, user click probability, social structures, and activity patterns on malware propagation in online social networks. We find that the highly skewed degree distributions and highly clustered structures shown in many social networks are instrumental in spreading the malware quickly at its early stage, but the highly skewed distribution of activities among online social network users and short active time spans by many online social network users slow down malware propagation. Also, the initial infection plays an important role in spreading malware, as malware starting from nodes that seldom get online do not lead to wide-scale infections. Moreover, using good social engineering techniques to fool online social network users into clicking malicious URLs with a high probability significantly accelerates malware propagation. **(3)** We study the effectiveness of *user-oriented defense* in which users spread alert messages to their friends once their machines have been found infected. We find that it is crucial to have a short detection period for such user-oriented defense schemes to be effective. We also notice that a more aggressive approach to spreading alert messages outperforms a relatively passive one in slowing down malware propagation only under certain circumstances. **(4)** We further analyze the effectiveness of *server-oriented defense* in which online social network websites deploy preventive schemes against malware spreading. We find it effective to slow down malware propagation by ensuring the cleanness of the top nodes with the most neighbors or active neighbors so that they do not send or receive malicious messages. By contrast, it is a *less* effective mitigation scheme to clean only the top most active nodes. Furthermore, we consider a preventive containment scheme which partitions the network into small "islands" and sanitizes messages that are delivered between these islands. We show that for this method to be effective, it is necessary to sanitize messages on a significant fraction of edges in the whole network.

The remainder of the paper is organized as follows. In Section 2, we present related work on malware propagation in social networks. In Section 3, we analyze the BrightKite dataset, including both the social structure formed by its users, user activity patterns, and the correlation between them. Section 4 briefly introduces our simulation environment and how we model malware propagation. Section 5 discusses the effects of some key factors on malware propagation, including initial infection, user click probability, social structure, and activity patterns. We study the effectiveness of user-oriented defense schemes in Section 6 and that of server-oriented defense in Section 7, respectively. We present the limitations of our work in Section 8 and draw concluding remarks in Section 9.

2. RELATED WORK

Malware propagation in social networks has been intensively investigated in the literature. Most of these works focus on analyzing the impact of social structures on epidemic spreading. Malware propagation in scale-free network structures has been studied in [14, 2, 9, 28]; malware propagation in small-world networks has been investigated in [34, 31, 23]. Malware propagation in specific types of social networks such as E-mail and IM networks has also been well studied in the past [24, 38, 20]. Common to these efforts, malware propagation is often analyzed with realistic network topologies but under simple assumptions on user activities. In the recent work by Faghani and Saidi [12], they investigated malware propagation in online social networks by using simulated topologies and user activities. In a recent work [36], Xu et al. proposed a correlation-based scheme to mitigate worm propagation in online social networks. Their work used the social graph data from the Flickr network but lacks realistic user activities. In our study, by contrast, we use not only realistic network topologies but also realistic user activities; our results reveal that user activity models play an important role in malware propagation, suggesting that future work on malware propagation in social networks should take into consideration not only network structures but also realistic user activity models.

Recently there have been a few efforts on analyzing human activities in online social networks. Benevenuto et al. analyzed user activities at four popular online social networks and provided unique insights on how users behave in online social networks [1]. Wilson et al. analyzed interactions among Facebook users in [35] and found that the activity graph formed by online social network users differ significantly from the social graph formed by their friendships. Their results agree well with our conclusion from this work that realistic activity models should be applied to study malware propagation in online social networks. Guo et al. found that contents generated by online social network users obey a stretched exponential distribution [15], which has been confirmed by our analysis on the BrightKite dataset. Our work differs from this line of research by focusing on exploring how user activity patterns affect malware propagation in online social networks.

3. ONLINE SOCIAL NETWORK ANALYSIS

Our analysis of online social networks is based on BrightKite [3], a location-based online social network that has a modest number of users. Typical online activities of a BrightKite user include location update, note posting, and photo sharing, and these can be done through a variety of interfaces, including Web, SMS, and Emails. Li and Chen have analyzed the social graph formed from the friendships of BrightKite users and also their mobility characteristics in [18]. Although some analysis performed in this work bears certain similarity with theirs, here we focus on aspects of online social networks that affect malware propagation. In any case, we shall state their results explicitly in the following discussion when necessary.

Our analysis uses the BrightKite dataset that was collected from March 21, 2008 through September 24, 2009, compared with the only two-month period analyzed in [18]. The BrightKite website pushes users' activity event such as a user checkin, a location update, a photo upload, or a note posting to a data distribution service called GNIP [13]. Each activity event is associated with a timestamp indicating when it was performed. We query these activities every day after midnight. The dataset we use has 65,770 users, who contributed more than 45 million activity events in total. The only data that can possibly be missing from this dataset is those activity events that users posted earlier but thereafter removed on the same day before we obtained them from GNIP. We believe that such events were rare and their impact on this study can thus be ignored. As BrightKite introduced bi-directional friendships only lately, we obtain a social graph represented as a directed graph instead of an undirected one used in [18].

As the goal of this work is to study malware propagation in general online social networks, one may question the generality of the data collected from a specific online social network like BrightKite. To address this issue, we analyze both the structure of the BrightKite social graph and its users' activity patterns and the results reveal that many properties of the BrightKite online social network have been observed in other online social networks. This suggests that BrightKite is indeed a typical online social network and our observations made about malware propagation in it can be translated into other social networks as well. Next, we shall present detailed analysis of the BrightKite network.

3.1 Friendship analysis

We first analyze the social graph formed by BrightKite users from a graph-theoretical perspective.

A scale-free network? Many online social networks have been found to be scale-free [22], which indicates that their degree distributions obey a strong power law. Figure 1 depicts the log-log plot of the out-degree distribution of the BrightKite users. We are more interested in the out-degree distribution than the in-degree distribution because a worm propagating in online social networks needs to obtain outbound contacts to spread itself. We use the method proposed in [6], which combines maximum likelihood estimation and Kolmogorov-Smirnov statistics, and derive the best power-law fit as one with cutoff $\beta = 68$ and scaling exponent $\alpha = 2.92$. The p-value of the model fitting is 0.24. As it is higher than the suggested threshold 0.1 [6], the power-law distribution seems to be a reasonable fit for the number of outbound friends in the dataset. It is noted that the scaling exponent we obtained is much higher than 1.9, which is the value derived in [18] with the linear regression method. The difference results from the fact that the intended model in our analysis has an extra cutoff parameter.

The implications of a scale-free structure on malware propagation in Brightkite-like online social networks are two-folds. First, as a power-law degree distribution suggests that a few nodes in the network have high degrees, succeeding in infecting these nodes at the early stage of the worm propagation can significantly speed up worm spreading [38]. Second, as the scaling exponent is less than ~ 3.4875, it is possible to have an epidemic that infects almost every vulnerable node in scale-free networks like BrightKite [14]. This can be confirmed from Figure 2, which depicts the sizes of connected components against their ranks in the social graph formed by BrightKite users. The sizes of the largest two connected components have 57,417 and 48 users, respectively. Here, a connected component refers to the set of nodes that are connected through edges. As in the graph edges are directed, we found that in the largest connected component, 44 users cannot be reached by the majority of the users and 799 users cannot reach the majority of the users. Hence, a malware starting from a single infection point can infect machines used by up to 87% of BrightKite users.

A small-world network? A small-world network is highly clustered and has a small characteristic path [34]. To measure how nodes in the BrightKite graph are clustered together, we calculate its *clustering coefficient*. Given a *directed graph* $G(V, E)$, the clustering coefficient of a node v is defined as $\frac{H_v}{n_v(n_v-1)}$, where H_v is the number of edges between the neighbors of node v and n_v gives the number of neighbors that node v has. The clustering coefficient of a graph is the average clustering coefficient over all its nodes. For the giant component of the BrightKite graph, its clustering coefficient as 0.091, which is slightly higher than 0.0782, the clustering coefficient of the early counterpart [18]. For a random graph generated from the Erdos-Renyi model with a similar number of nodes and edges, the clustering coefficient is only 0.000156, which is much smaller than that of the BrightKite graph.

The characteristic path length of a graph is defined as the average length of the shortest paths of all node pairs. Recall that 44 users cannot be reached by the majority of the users and 799 users cannot reach the majority of the users in the largest connected component of the BrightKite graph. We thus found that 1.67% of node pairs do not have a legitimate path. For the remaining node pairs, the average shortest path has 5.768 hops, which agrees well with the six-degrees-of-separation phenomenon observed in other social networks [33]. The histogram of the shortest path length among routable path lengths is given in Figure 3. The network diameter is 18 hops, which is higher than what was observed in the earlier BrightKite graph. On the other side, if it is a random graph generated from the Erdos-Renyi model with a similar number of nodes and edges, the fraction of node pairs with legitimate paths is 99.93%, among which the average shortest path has 6.5 hops (over 20 runs).

Regarding the giant component of the BrightKite graph, as its average shortest path length is comparable to that in the random graph and its clustering coefficient is much smaller than that in the random graph, it closely reflects a small-world network structure [34], although a small fraction of node pairs in it do not have a path in between. Given that small-world networks are highly clustered and have small characteristic path lengths, epidemic disease can spread easily and quickly to the entire population, instead of being limited within small regions in the network. This suggests that from a structural perspective, it is not easy

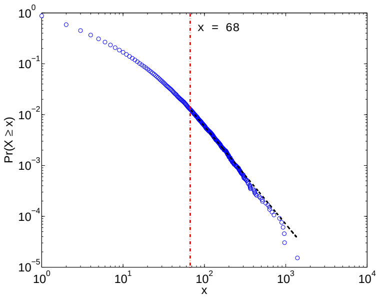


Figure 1: The log-log plot of the out-degree distribution and power-law fitting

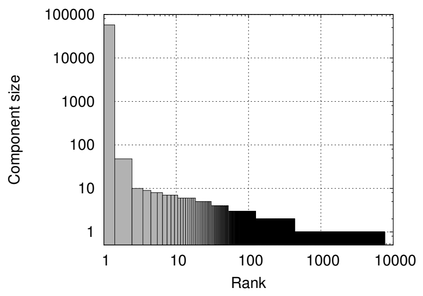


Figure 2: Sizes of connected components against their ranks in the social graph

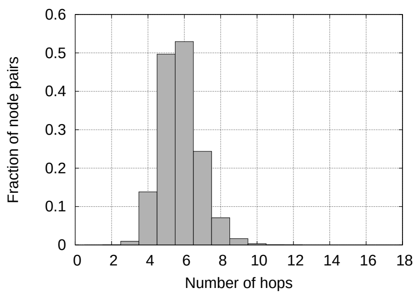


Figure 3: Histogram of the lengths of the shortest paths among node pairs

to mitigate malware propagation in BrightKite-like online social networks.

3.2 Activity analysis

Besides the network structure, users' online activities can also affect worm propagation. Even a well-connected user, if he rarely gets online, poses little impact on malware propagation in the network. In this section, we analyze activity patterns of BrightKite users that potentially affect malware propagation. It is noted that among the 65,770 BrightKite users, only 58,455 of them have performed at least one activity event. We call these *active users*. In the following discussion, we are limited to only active users.

Number of activity events per user. In Figure 4, we depict in a log-log form the number of activity events (i.e., location update, photo upload and note posting) that were performed by each user against his rank in the dataset. The largest and average number of activity events generated by an active user is 9,684 and 77.8, respectively. It is noted that the average number of activity events generated by an active BrightKite user is higher than what it was observed in [18], suggesting that BrightKite users use the online social network more frequently than before.

Recently Guo et al. found that contents generated by online social network users obey a stretched exponential distribution, whose CCDF (Complementary Cumulative Probability Distribution Function) is given as: $\mathbb{P}(X \geq x) = e^{-(x/x_0)^c}$ [15]. Its rank distribution function can be derived as $y_i^c = -a\ln(i) + b$, where y_i denotes the number of contents generated by the user of rank i. It would be interesting to see whether the activity events generated by active BrightKite users also follow this pattern. It is obvious to see that a power-law distribution is not a good fit because the log-log plot is not linear. Using the maximum likelihood estimation method proposed in [15], we obtain the parameters in the rank distribution function as follows: $c = 0.341927$, $a = 2.21$, and $b = 24.879$. Using Pearson's chi-square test, we have confirmed that the derived stretched exponential distribution is indeed a good fit.

Stretched exponential distributions imply that online activities are not as dominated by a small group of top users as in a network with power-law distributions [15]; that is to say, online activities are more stretched out over the entire population in a stretched exponential network. If we assume that the number of activity events is a good indication of how frequently a user visits the online social network, having online activities more spread out over the entire population is instrumental to malware propagation in online social networks because vulnerable machines become infected only when users get online and click on malicious links sent from their friends.

Active time span per user. A common phenomenon in online social networks is that some users lose interests after trying a few times and become inactive. This is also observed in the BrightKite dataset. We define the *active time span* of a user as the duration between her first and last activity events in the dataset. Figure 5 gives the CCDF of the active time spans of the active BrightKite users. We note that as many as 35% of the active BrightKite users actually have an active time span no greater than 10 days. Hence, machines used by these users, even if vulnerable to a malware attack, may not be helpful in spreading the malware due to their inactivity.

At any given time, the number of infective machines in the online social network is bounded by the number of machines used by active users. Figure 6 presents the number of active users that perform at least one activity event in a day. We conjecture that the dropoff at the tail was because the data was not collected completely from the BrightKite server. Later in trace-driven simulation, we do not use the activity events during this period. Interestingly, we observe that there is a surge in the middle of the graph. As this surge coincided with the release date of the iPhone version of BrightKite, we conjecture that iPhone users contributed a significant portion of BrightKite activities after the surge. This means that to spread malware in the BrightKite online social network, it would pay off if the malware could exploit some vulnerabilities on iPhones. It is also evident that BrightKite user activities exhibit strong weekly patterns. For instance, there are usually more BrightKite users that perform activity events on Saturday than the other days in a week. Hence, it would help the malware to spread quickly in the beginning if it is released on Saturday.

3.3 Friendship-activity correlation

After analyzing the friendships and activities in the BrightKite dataset, an interesting question would be: is there any strong correlation between the number of friends a user has and how active he is in the online social network? Figure 7 depicts the relationship between the number of activity events generated by a BrightKite user and the number of her contacts. We further use the correlation coefficient to measure the linearity of this correlation. Define $X = \{x_i\}_{i=1,\ldots,n}$, where x_i denote the number of outbound friends user i, and

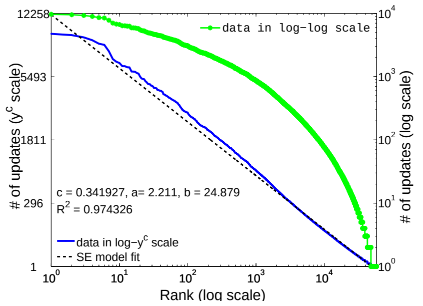


Figure 4: Stretched exponential distribution of user updates

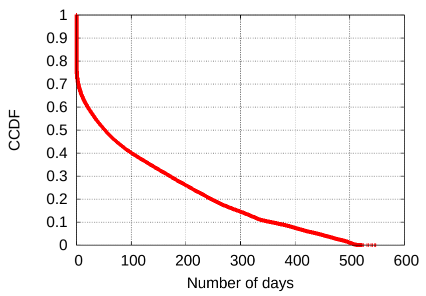


Figure 5: The CCDF of the active time spans of the BrightKite users

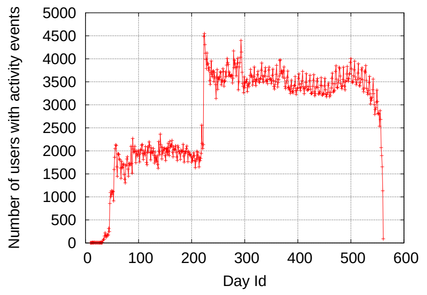


Figure 6: Number of users with activity events on each day in the dataset

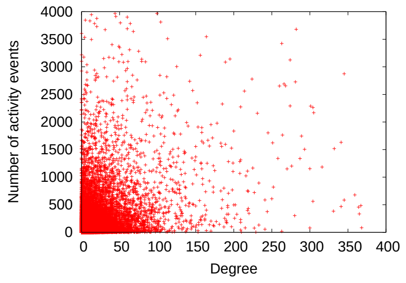


Figure 7: Number of activity events vs. node degree

$Y = \{y_i\}_{i=1,\ldots,n}$, where y_i denote the number of activity events generated by user i. The correlation coefficient is defined as follows:

$$c(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{X})(y_i - \bar{Y})}{[\sum_{i=1}^{n}(x_i - \bar{X})^2]^{1/2}[\sum_{i=1}^{n}(y_i - \bar{Y})^2]^{1/2}},$$

where $\bar{X}$ and $\bar{Y}$ are the mean of X and Y, respectively. If $c(X,Y) > 0$, X and Y are positively related; if $c(X,Y) < 0$, X and Y are negatively correlated; otherwise, there is no correlation between X and Y. If $|c(X,Y)|$ is closer to 1, the correlation between X and Y is stronger and $|c(X,Y)| = 1$ means that X and Y have an exact linear relationship.

For the BrightKite dataset, we find that the correlation coefficient between X and Y is 0.3809, which is positive but much smaller than 1. Hence, the number of friends a BrightKite user has is positively correlated with how active he is in the network, although only in a weak sense. This observation may affect the design of intelligent malware. For instance, if a malware aims to evade detection by limiting the number of messages it generates in the network [37], it may want to let each infected machine contact only a small number of neighbors (or friends). As the number of friends a user has is only weakly correlated with his online activity intensity, attempting to infect the most popular neighbors may not always be a wise decision.

4. TRACE-DRIVEN SIMULATION

In the previous section, we have statically analyzed the BrightKite dataset from both friendship and activity perspectives. Although such analysis sheds light on implications of structural and behavioral properties on malware propagation in online social networks, we are still unclear on the dynamics of malware propagation in online social networks, such as how long it takes to infect the majority of the vulnerable population. Answers to such questions are instrumental for us to prepare effective defense schemes against malware propagation in online social networks.

Ideally, we wish we could analyze traces of real-world malware propagation in online social networks such as Facebook [11] and Twitter [32]. Like previous measurement studies on Internet worms and botnets, the first-hand data offer tremendous insights into operational dynamics of these malware in the reality. However, such measurement data related to malware propagation in online social networks are still not available. Hence, we resort to using simulation to study malware propagation dynamics in scenarios that capture important details of the real-world counterpart. To be more realistic, we build a malware propagation simulator that is directly driven by the BrightKite dataset, such as the social graph formed from the user friendships and activity events generated by each user. In addition to BrightKite users, we also simulate the BrightKite server, which keeps messages until they are fetched by the recipients.

In the simulation, when the machine used by a client is infected, it sends a message embedded with a URL that points to a malware server that hosts the malicious code to each of the client's friends in the social graph. The message is first delivered to the online social network server after an end-to-end delay drawn from distribution $\mathcal{D}$; on the arrival of the message, the server keeps the message locally until it is fetched by the recipient.

A client's machine becomes online whenever there is an activity event in the BrightKite dataset. When it becomes online, it sends a request to the online social network server and the server sends back a list of messages that are destined to the client. The end-to-end delay that a message experiences is also generated from distribution $\mathcal{D}$. When a client receives a list of incoming messages, if there is a URL link in it, she clicks on it with probability p. The thinking time before clicking on it follows distribution $\mathcal{T}$. After a client clicks on a malicious URL, a request is sent to the malware server, which in return sends back the malware code. The end-to-end delay between the client's machine and the malware server is also randomly drawn from distribution $\mathcal{D}$. The transmission delay of the malware code by the malware server is drawn from distribution $\mathcal{R}$. When the malware code arrives at the client machine, if it is not infected yet, the machine gets infected and the above process repeats.

The propagation model we described above is essentially the classical SI (Susceptible-Infective) model. It is worthy noting the difference between it and the SIR (Susceptible-Infective-Removed) which takes into cosideration node recovery from an infected state in the context of online social network worms. For traditional Internet worms like Code Red [8] and Slammer [29], machines infected by these worms often perform scanning continuously to look for new vulnerable machines for infection. Hence, cleaning these infected machines into a healthy state slows down malware propagation, as it leads to fewer infective machines that search for new victims. By contrast, malware propagation in online social networks uses a different approach for spreading: when a user account is compromised, a malicious message is sent (or broadcast) to each of her friends, irrespective of whether her friends are online or not. In our work, we do not consider that cleaned nodes are reinfected and then send malicious links again. Consequently, whether an infected machine is cleaned or not later, it will not change the course of malware propagation any more. Given these observations, we adopt the simple SI model first when we study the propagation dynamics of online social network worms in Section 5 as the SIR model would only change the total number of infectives in the network but *not* the dynamics of the malware propagation process. In Section 6, we shall continue to study approaches that combine both the SIR model and some warning mechanisms to slow down worm propagation in online social networks.

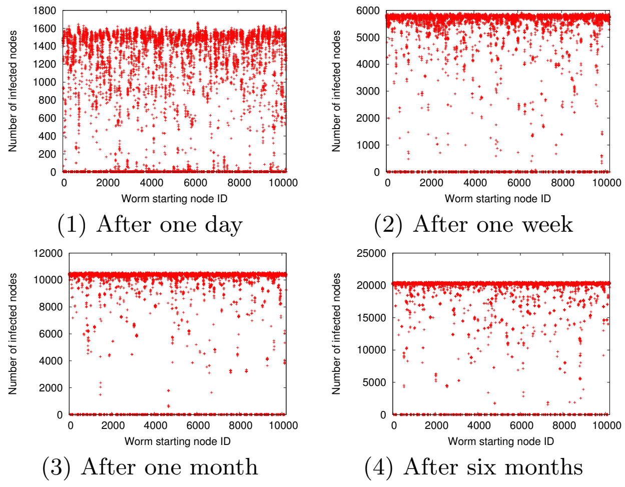
5. PROPAGATION DYNAMICS

In this section, we use trace-driven simulation to study the impact of initial infection, user click probability, social graph structures, and activity patterns on malware propagation. We choose end-to-end delays form the empirical distribution of packet delays collected from traces by CAIDA [4]. The thinking time $\mathcal{T}$ before a user clicks on a URL (or decides not to) is drawn from a normal distribution with both mean and standard deviation as 10 seconds, and the transmission delay of the malware code is uniformly distributed between 5 and 15 seconds. Although these assumptions may not reflect real network conditions, they pose little impact on the propagation dynamics of online social network malware because their time scales are much smaller than those of BrightKite users' activity events. For a similar reason, we ignore normal user messages.

5.1 Initial Infection

We first analyze the effect of initial infection on the propagation dynamics. In all the experiments, we assume that each user clicks on an embedded URL with probability 0.5. We choose the initial infection as follows: we divide all the users in the BrightKite dataset into k groups, where users in group i ($0 \leq i \leq k-1$) have $[2^i, 2^{i+1})$ outbound friends. From each group, we randomly choose 50 users as the initial infection point (if there are less than 50 users in a group, we simply choose all the users). For each infection point, we simulate malware propagation 20 times with different seeds to generate random numbers. In total, we have 10,200 sample runs spread over 13 groups.

Figure 8 depicts the number of infections after one day, one week, one month, and six months against different initial infections. Each sample run corresponds to a point in the figure. From Figure 8, we observe that as time passes by, the number of infections tends to fall into two extremes, far from being evenly distributed. This is further verified as follows. For each measurement time t (one day, one week, one month, and six months after initial infection), we obtain the largest number of infections $I_t^{(max)}$ over all 10,200 cases.



(1) After one day (2) After one week

(3) After one month (4) After six months

Figure 8: Number of infections in each sample run under different initial infection points

The ratio of the number of infections at time t in each case to $I_t^{(max)}$ falls into five different bins: less than 20% (Bin 1), between 20% and 40% (Bin 2), between 40% and 60% (Bin 3), between 60% and 80% (Bin 4), and between 80% and 100% (Bin 5). Figure 9 depicts the number of cases in each bin. Clearly, at each measurement time, most of the cases belong to either Bin 1 or 5. This phenomenon agrees well with other types of Internet worms: once the infection takes off, it does not take long for malware to infect the majority of the entire vulnerable population.

For ease of explanation, we say that a sample run (or case) leads to *rare infections* if the number of infections is no more than 10. For each measurement time, we remove the cases with rare infections. The following table shows the mean, standard deviation (std), and coefficient of variation (COV) of the remaining cases for each measurement time:

	1 day	1 week	1 month	6 months
mean	1274.45	5534.34	10157.78	19686.02
std	368.20	668.88	926.05	2085.83
COV	0.2889	0.1209	0.0912	0.1060

The table shows that for all four measurement times, the COV is much smaller than 1. Hence, without considering those cases with rare infections, there is low variation among remaining ones.

We further explore what properties of initial infection points affect the malware propagation speed. As the initial phase is crucial to malware spreading, we analyze the cases still with rare infections after six months. In Figure 10(1), we plot the number of sample runs with rare infections after six months against the degree of the initial infection point. We observe that although a low degree tends to cause more sample runs with rare infections, infecting a node with a number of neighbors initially does not necessarily cause a significant number of infected nodes after six months. It is noted that the average number of neighbors in the BrightKite dataset is about 8. For one initial infection point with as many as 38 neighbors, 15 of the 20 runs do not lead to a significant number of infections after six months.

After examining the cases where the initial infection point has a high degree but leads to only a few infections after 6 months, we find that in these cases, although the initial in-

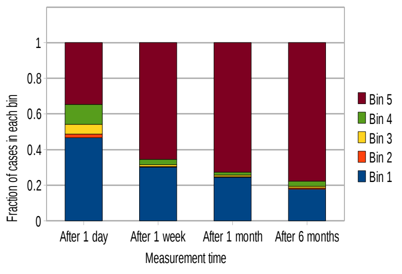
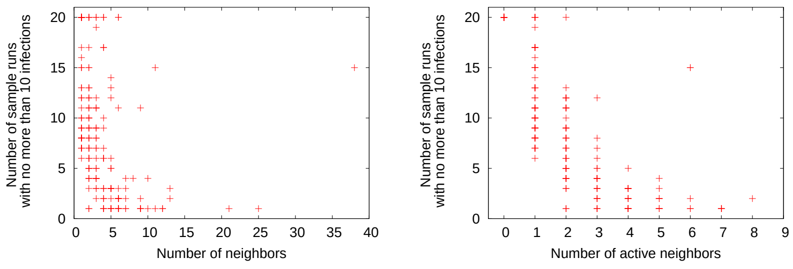


Figure 9: Fraction of cases in each bin



(1) Degree (2) Active neighbors

Figure 10: Number of cases vs. degree and active neighbors with rare infections after 6 months

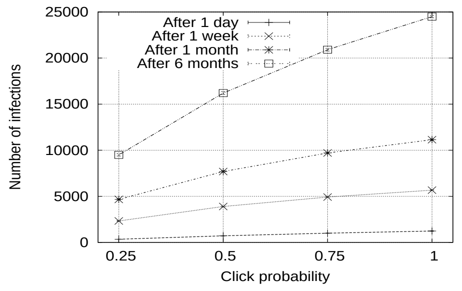


Figure 11: Impact of click probabilities

fection point has a number of neighbors, only a few of them were active within the simulated period. In Figure 10(2), we depict the number of sample runs with rare infections after six months against the number of neighbors that were active at least once within the six months. It is clear that with more active neighbors, it is more likely to spread the malware beyond the initial infection point. From the attacker's perspective, his choice of the initial infection point should take into consideration not only the number of neighbors it has but also the number of neighbors that may become active after the malware is released.

5.2 Click Probability

We vary the click probability among 0.25, 0.5, 0.75, and 1. For each click probability, we use the same set of initial infection points as in Section 5.1 and thus have 10,200 sample runs for each click probability after simulating each case for 20 times. In Figure 11, for each click probability, we present the mean number of infections at four measurement times. Unsurprisingly, with a higher click probability, the malware spreads to a higher number of nodes at each measurement time. This suggests that using tricks to fool people into clicking embedded URLs with a high probability indeed pays off to accelerate malware propagation.

5.3 Social Graphs and Activity Patterns

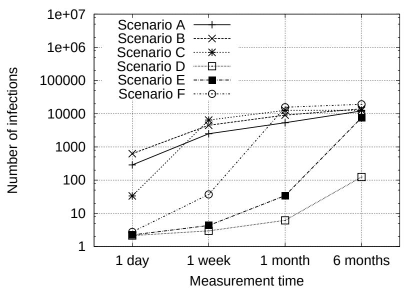
To understand the impact of different aspects of online social networks on malware propagation, we use different social graphs and activity models in our experiments. We consider two types of social graphs: the first one (**BrightKite graph**) is the giant component of the BrightKite social graph, and the second one (**Erdos-Renyi graph**) the Erdos-Renyi social graph with a similar number of nodes and edges. The giant component of the BrightKite graph has 57,417 nodes and 460,503 edges, and it produces 1,759,026 activity events in six months. It is noted that among all the 57,417 nodes, only 29,323 of them have been active at least once. Regarding the activity models, we use three different methods. In the first method (**BrightKite activities**), we directly use the set of activity events produced by the giant component of the BrightKite graph. In the second method (**locally randomized activities**), we keep the number of activity events generated by a user intact but the time of each activity event is uniformly drawn from the simulated interval, which lasts six months. In the last method (**globally randomized activities**), we randomly choose 29,323 nodes and each of them produces 60 activity events. The time of each of these activity events is uniformly drawn from the simulated interval. Note that the three activity models generate about the same number of activity events in each simulation run. In the third method, we only choose 29,323 nodes because only active users can possibly be infected. With different combinations of social graphs and activity models, we have the following six scenarios:

Scenario	Social Graph	Activity Model
A	BrightKite	BrightKite
B	BrightKite	Locally randomized
C	BrightKite	Globally randomized
D	Erdos-Renyi	BrightKite
E	Erdos-Renyi	Locally randomized
F	Erdos-Renyi	Globally randomized

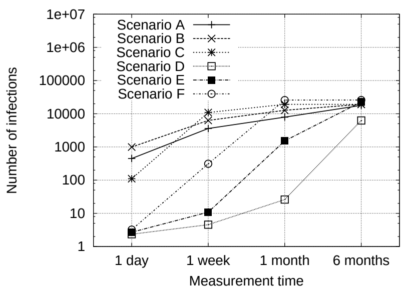
In each scenario, we randomly choose the initial infection point among all the nodes in a normal scale. Note that we randomly choose the initial infection point in a logarithmic scale previously. For each scenario, we also vary the click probability among 0.5, 0.75, and 1.

For each scenario, the average number of infections at different measurement times is shown in Figure 12. From the simulation results, we can make the following observations. First, under the BrightKite graph, the malware propagates faster than it under the Erdos-Renyi graph at its early stage. This is because the BrightKite graph is highly clustered and it is easy to spread the malware onto nodes that are highly connected. Particularly, with the BrightKite graph and the original activity events, the malware can infect hundreds of nodes after one day, regardless of the click probability, but irrespective of the activity model used, only a few can be infected after one day under the Erdos-Renyi social graph.

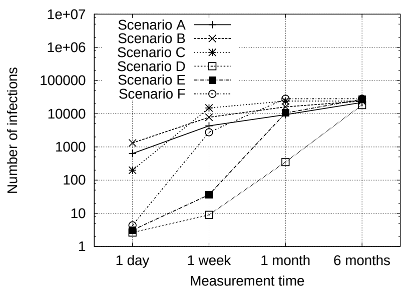
Second, compared with the original activity events, the locally randomized activity model helps accelerate malware propagation, regardless of the click probability and the social graph. From Figure 5, we know that many users actually have a short active time span. For these users, if they receive



(1) Click probability = 0.5



(2) Click probability = 0.75



(3) Click probability = 1

Figure 12: Number of infections under different social graphs and activity models

a malicious URL after they become inactive, their machines cannot be infected. The locally randomized activity model spreads activity events of each user over the entire simulation period, which helps speed up malware propagation.

Third, the globally randomized activity model further improves the malware propagation speed, compared with the locally randomized activity model. This is because the globally randomized activity model spreads activity events not only over the entire simulation period but also among different active users. Recall that the activity events in the BrightKite dataset can be well characterized by the stretched exponential model. Although not as skewed as a power law distribution, this model still produces uneven activity event distribution: for the giant component of the BrightKite social graph, the top 20% users actually contributed 94.7% of those 1,759,026 activity events. With more online activities, those nodes with only a small number of activity events in the original BrightKite dataset would more likely get infected or get infected at an earlier time.

It is however noted that the globally randomized activity model slows down the malware propagation at its early stage under the BrightKite social graph, irrespective of the click probability. The propagation speed of the malware at its early stage depends on how quickly the malware can infect those highly connected nodes. From the friendship-activity correlation analysis in Section 3.3, we know that the number of friends a BrightKite user has is positively correlated with the number of activity events she generates, suggesting highly connected nodes tend to be more active than other nodes. The globally randomized activity model, however, destroys such a positive correlation: regardless of how many friends a user has, her activity level is the same as others. As a node gets infected only when it gets online, the likelihood that the malware can infect a highly connected node under the globally randomized activity mode is lower than that under the original BrightKite activities, which thus hinders the malware from spreading quickly at its early stage.

Fourth, if the click probability is as low as 0.5, the malware spreading under the Erdos-Renyi social graph and the original set of activity events infects only 0.22% of all the nodes even after six months. By contrast, replacing the Erdos-Renyi social graph with the BrightKite social graph can infect 22% of all the nodes after six months. This further confirms that the social structure of users in online social networks can significantly impact malware propagation in such networks.

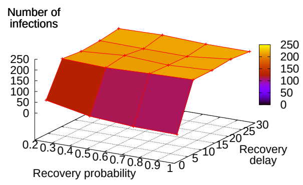
Fifth, under the most realistic scenario (Scenario A), when the click probability is 1, the malware eventually infects about 79% of possible nodes (i.e., active users during the six month period). Among all the 200 sample runs, the largest number of infections after six months is about 92% of all active users. Hence, even if the malware chooses a good infection point and adopts a social engineering technique that could fool the users into clicking embedded URLs with a high probability, there is still a small fraction of active nodes that are difficult to infect. To further increase the infection coverage, the malware needs to use other techniques, such as using multiple infection vectors rather than only relying on the online social network for its spreading.

6. USER-ORIENTED DEFENSE

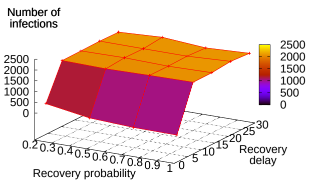
In the previous section, we have analyzed the factors that affect malware propagation in online social networks. In this and the next sections, we will investigate different defense strategies to slow down such worms. More specifically, we focus on user-oriented defense in this section and server-oriented defense in the next one. In the user-oriented defense, when an infected machine recovers, it sends a warning message to each of its neighbors. When a neighbor gets online and receives such a message, it becomes immune to the worm infection. Moreover, if the neighbor chooses to forward the warning message further to her neighbors, we say that it is an *active* defense mode; otherwise, it is a *reactive* defense mode.

In the new set of experiments, we vary the recovery probability of an infected machine among 0.25, 0.5, 0.75, and 1.0. The recovery delay is drawn from a uniform distribution in a range of one day; the mean of the recovery delay is chosen among 1 day and 1-4 weeks. We also consider both the defense modes. For each scenario, we randomly choose 200 initial infection points and for each of them, we simulate 10 times. In total, we have 80,000 simulation runs. In all these experiments, we let each user's click probability be 0.5.

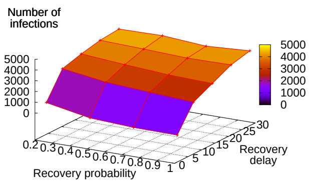
Figure 13 depicts the average number of infections in the network for each scenario after one day, one week, one month, and half a year. In the baseline case where no user-oriented defense scheme is deployed, the average number of infections after after one day, one week, one month, and half a year is 286.8, 2495.0, 5361.8, and 12164.5, respectively. The key observations made from these results are summarized as follows. **(i)** Increasing the recovery probability helps slow down malware propagation. But the mitigation effect is much more prominent under the reactive defense mode than the active defense mode. This is because the active defense mode pushes warning messages automatically as long as at least one user triggers this process. By contrast, under



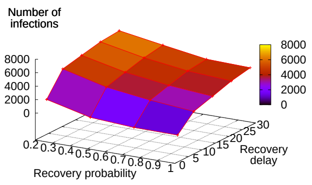
(1) Reactive, one day



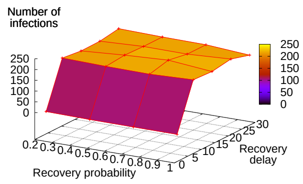
(2) Reactive, one week



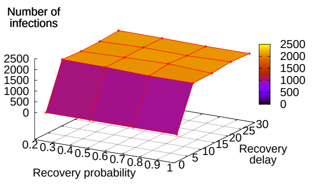
(3) Reactive, one month



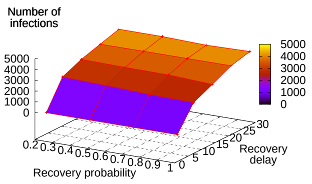
(4) Reactive, six months



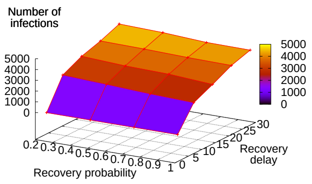
(5) Active, one day



(6) Active, one week



(7) Active, one month



(8) Active, six months

Figure 13: Number of infections under user-oriented defense

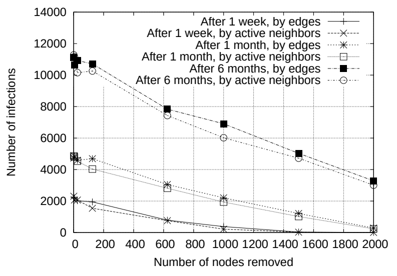
the reactive defense mode, warning messages are sent out only by recovered machines; hence, the recovery probability has more impact on the malware propagation process. **(ii)** When the discovery delay is short (i.e., only one day), malware spreading slows down significantly, regardless of the defense mode. By contrast, when the discovery delay is long, the speed of malware propagation is not affected during the short period after the worm starts spreading, as evidenced by the plateau shown in Figures 13(1), (2), (5), and (6). But in the long run, reducing the discovery delay still helps mitigate the malware propagation. **(iii)** The active defense mechanism outperforms the reactive couterpart when the recovery probability is relatively low and the recovery delay is also short. They perform similarly under other scenarios because when the recovery delay is short and the recovery probability is high, the reactive defense scheme is also effective in slowing down malware propagation; when the recovery delay is too long, the active defense scheme cannot prevent the early takeoff of the worm spreading either (although the active defense scheme still performs better when the recovery probability is low and the recovery delay is large). **(iv)** Compared against the baseline case where no defense is deployed, both reactive and active defense schemes help slow down malware propagation in the long term. Figures 13(7) and 13(8) bear little diference, suggesting that once the active defense scheme becomes effective, it indeed helps stop malware from spreading onto new hosts. By contrast, Figures 13(3) and 13(4) tell us that if the discovery probability is low under the reactive defense scheme, the malware can still infect a number of new victims after one month since the worm starts.

One important message we learn from Figure 13 is that a quick discovery period plays a crucial role in containing malware infection within a small range. If the active defense mechanism is deployed, malware spread can be controlled even if the discovery probability is low; otherwise if the reactive defense mechanism is applied, a high discovery probability is also necessary to contain malware successfully.

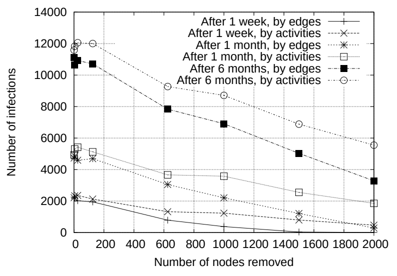
7. SERVER-ORIENTED DEFENSE

In the previous section, we have studied the effectiveness of user-oriented defense. We now go further to analyze the performance of defense schemes deployed by online social network servers. We consider online social networks where their central servers do not have the capacity to check every message containing URLs. A server thus selectively chooses a subset of messages to ensure that URLs included in them are not malicious. The maliciousness of a message can be tested by analyzing the behavior after clicking every embedded URL in a virtual machine. In this study, we first focus on a simple scheme in which the server chooses a subset of nodes and for every message delivered *from and to* it, the server ensures that no malicious URL is included. We say that the server *sanitizes* each of these nodes. We consider three sanitization schemes. In the first one (**by edges**), we rank all the nodes by their degrees and choose the top k nodes with the highest degrees to sanitize. In the second scheme (**by active neighbors**), we rank all the nodes by the number of active neighbors that they have and choose the top k nodes with the most active neighbors to sanitize. In the third scheme (**by activities**), we rank all the nodes by the number of activity events generated by them and choose the top k nodes with the most activity events to sanitize. Once a node is sanitized, it does not get infected even if the user clicks on a malicious embedded URL.

In a new set of experiments, we let k be chosen from $\{1, 5, 25, 125, 625, 1000, 1500, 2000\}$ and for each case we perform 200 sample runs. The user click probability is set to be 0.5. The effectiveness of the three sanitization schemes is illustrated in Figure 14. We note that the scheme by active neighbors performs slightly better than the one by edges, especially when there are a significant number of nodes removed. When the number of removed nodes is less than 125, these two schemes are almost equivalent because at most one node differs. As explained in Section 5.1, the number of active neighbors is more important to malware propagation because it dictates the maximum number of neighbors that a node can infect. Another interesting observation is that sanitization by activities performs the worst among the three schemes. This is not surprising because according to Figure 7, nodes with a large number of activity events do not necessarily have many neighbors, and the importance of a node in spreading the malware highly depends on how many active neighbors it has.



(1) Edges vs. active neighbors



(2) Edges vs. by activities

Figure 14: Number of infections under different sanitization schemes

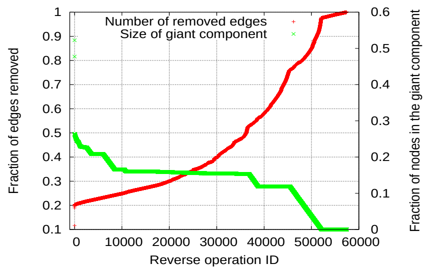


Figure 15: Sizes of giant components under preventive containment

From Figure 14, we also observe that in the range considered, the number of infections seems to decrease only linearly with the number of nodes removed, regardless of the sanitization scheme. This is counterintuitive because one might think that removing the top few connected nodes would slow down malware propagation more significantly than the other nodes; if that were true, the curve would be sublinear. On an encouraging note, selectively sanitizing the nodes with the highest numbers of either edges or active neighbors can slow down malware propagation significantly. In [38, 27], it has been observed that selective or targeted immunization (which is similar to sanitization discussed here) is an effective approach to stopping malware propagation in scale-free networks. Our results confirm that that conclusion holds under realistic activity models for online social networks.

In the following discussion, we consider another defense scheme called *preventive containment*. Its basic idea is that we partition the network into a number of "islands" and sanitize the messages that are delivered among these islands to ensure that they are not malicious. Hence, malware starting from a node inside an island is contained within that island. For preventive containment to be effective, we need to ensure that the size of each island is small. To partition the BrightKite network, we use a hierarchical community detection algorithm to produce a dendrogram, which shows how nodes are clustered together to form communities. As nodes in the same community are supposed to be highly clustered and nodes among different communities are only sparsely connected, our intuition is that removing those inter-community edges would help break down the connectivity of the whole graph.

We used the community structure detection algorithm by greedy optimization of modularity in the igraph library [16], whose implementation was based on the work in [5]. The algorithm works in a bottom-up fashion: it starts with individual nodes, forms small communities from them, and then iteratively add edges between smaller communities to form larger communities until the whole graph is generated. The whole process produces a hierarchical dendrogram. Based on this dendrogram, we remove edges in a top-down fashion and in a reverse order until the whole graph becomes individual nodes. We call an *operation* a procedure of removing edges between two communities during this process.

After each operation, we measure the total number of edges that have been removed and the size of the giant component in the graph so far. We start from the giant component of the BrightKite graph initially and the results are illustrated in Figure 15 after each operation. Clearly, when we keep removing edges from the graph, the size of the largest component in the remaining graph decreases. The results, however, cast a pessimistic note on the effectiveness of preventive containment. For instance, in order to contain the malware propagation within 52%, 26%, 20%, 15%, and 10% of the nodes in the giant component of the BrightKite graph, it is necessary for us to ensure the sanity of the messages on 12%, 20%, 23%, 53%, and 78% of the edges, respectively. That is to say, to contain malware spreading within a small number of nodes once it starts from even a single node, we need to sanitize messages on a *significant* number of edges.

8. SCOPE OF OUR WORK

In this work, we use trace-driven simulation to study malware propagation in online social networks. Our conclusions can only be as accurate as their underlying assumptions. We assume that each BrightKite user becomes active only after she performs some activities visible to the system, such as user checkin, location update, photo uploading, and note posting. This may not be true because some users may remove their activity events on the same day before we had a chance to download these events. Machines used by these users may still get infected if they click on some malicious URLs sent from their friends. Albeit such deleted events are believed to be rare, the experiments in this work may underestimate the malware propagation speed.

Another limitation is that we assume each account name in the BrightKite dataset corresponds to a unique machine owned by a BrightKite user. This may not be true because a BrightKite user can have multiple accounts or use multiple machines to access her account. In the former case, our simulation experiments would overestimate the number of nodes in the social graph, while in the latter one, the number of nodes in the experiments is underestimated.

Moreover, the user click probability in our experiments is the same over all the users. This is not realistic given the fact that different online social network users typically have different awareness levels on malicious embedded URLs sent from their friends. Also, the level of trust that an online social network user gives to a message also depends on her relationship with the sender. Hence, the response of an online social network user to URLs embedded in the messages is simplified in our simulation experiments.

Even with these limitations, our work still sheds light on the characteristics of malware propagation in realistic online social networks. The goal of this work is not to draw definite conclusions on how fast a malware can propagate in a real-world online social network. Given the high dynamics of online social networks these days (e.g., increasing number of users), providing an exact answer to that question is surely beyond our scope. Instead, our aim is to investigate the effects of different aspects of online social networks on malware propagation, especially those due to user activities, which were rarely studied in previous works.

9. CONCLUSIONS

Malware specifically targeting online social networks are on the rise. In this work, we aim to study the characteristics of malware propagation in online social networks. We analyze a real-world location-based online social network, including the social structure formed by its users and its user activity patterns. We further use trace-driven simulation to study the impact of initial infection, user click probability, social structures, and user activity patterns on malware propagation in online social networks, and offer insights on how to defend against malware attacks in such networks.

10. REFERENCES

[1] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida. Characterizing user behavior in online social networks. Chicago, Illinois, USA, 2009.

[2] L. Briesemeister, P. Lincoln, and P. Porras. Epidemic profiles and defense of scale-free networks. In *Proceedings of the 2003 ACM workshop on Rapid malcode*, Washington, DC, USA, 2003.

[3] `http://www.brightkite.com/`.

[4] `http://www.caida.org`.

[5] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(6), 2004.

[6] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. arXiv:0706.1062, June 2007.

[7] `http://www.theregister.co.uk/2010/06/01/facebook_clickjacking_worm/`.

[8] `http://www.cert.org/advisories/CA-2001-19.html`.

[9] Z. Dezső and A. Barabási. Halting viruses in scale-free networks. *Physical Review E*, 65(5):055103, May 2002.

[10] `http://www.facebook.com/press/info.php?statistics`.

[11] `http://www.computerworld.com/s/article/9128842/Koobface_worm_to_users_Be_my_Facebook_friend`.

[12] M. R. Faghani and H. Saidi. Malware propagation in online social networks. In *Proceedings of the 4th IEEE International Conference on Malicious and Unwanted Software*, Montreal, Canada, October 2009.

[13] `http://www.gnip.org/`.

[14] C. Griffin and R. Brooks. A note on the spread of worms in scale-free networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(1):198–202, Feb. 2006.

[15] L. Guo, E. Tan, S. Chen, X. Zhang, and Y. Zhao. Analyzing patterns of user content generation in online social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, Paris, France, 2009.

[16] `http://igraph.sourceforge.net/`.

[17] `http://www.kaspersky.com/news?id=207575670`.

[18] N. Li and G. Chen. Analysis of a location-based social network. In *Proceedings of the International Symposium on Social Intelligence and Networking*, 2009.

[19] `http://www.pandasecurity.com/NR/rdonlyres/BBB11FA2-10BD-435A-B936-5CD55C45E427/0/Malwaretargetssocialnetworks.pdf`.

[20] M. Mannan and P. C. van Oorschot. On instant messaging worms, analysis and countermeasures. In *Proceedings of the 2005 ACM workshop on Rapid malcode*, Fairfax, VA, USA, 2005.

[21] `http://www.bnonews.com/news/242.html`.

[22] 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*, 2007.

[23] C. Moore and M. E. J. Newman. Epidemics and percolation in small-world networks. *Physical Review E*, 61(5):5678–5682, May 2000.

[24] M. E. J. Newman, Stephanie Forrest, and Justin Balthrop. Email networks and the spread of computer viruses. *Physical Review E*, 66(3), 2002.

[25] `http://en-us.nielsen.com/main/news/news_releases/2009/september/nielsen_reports_17`.

[26] `http://www.statemaster.com/encyclopedia/Orkut`.

[27] R. Pastor-Satorras and A. Vespignani. Immunization of complex networks. *Physical Review E*, 65, 2002.

[28] R. P. Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Physical Review Letters*, 86(14):3200–3203, Apr 2001.

[29] `http://www.cert.org/advisories/CA-2003-04.html`.

[30] `http://news.cnet.com/8301-13577_3-10160850-36.html`.

[31] Telo and A. Nunes. Epidemics in small world networks. *The European Physical Journal B - Condensed Matter and Complex Systems*, 50(1):205–208, March 2006.

[32] `http://www.pcworld.com/article/162992/twitter_worm_attack_continues_heres_how_to_keep_safe.html`.

[33] D. J. Watts. *Six Degrees: The Science of a Connected Age*. W. W. Norton & Company, 2003.

[34] D. J. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[35] C. Wilson, B. Boe, A. Sala, K. P.N. Puttaswamy, and B. Y. Zhao. User interactions in social networks and their implications. In *Proceedings of the 4th ACM European conference on Computer systems*, Nuremberg, Germany, 2009.

[36] W. Xu, F. Zhang, and S. Zhu. Toward worm detection in online social networks. In *Proceedings of the 25th Annual Computer Security Applications Conference (ACSAC)*, 2010.

[37] G. Yan, Z. Xiao, and S. Eidenbenz. Catching instant messaging worms with change-point detection techniques. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, San Francisco, California, 2008.

[38] C. C. Zou, D. F. Towsley, and W. Gong. Email worms modeling and defense. In *Proceedings of the International Conference on Computer Communications and Networks*, 2004.