Malware Propagation in Online Social Networks

Mohammad Reza Faghani
Department of Computer Science and Engineering
York University, Canada

Hossein Saidi
Department of Electrical and Computer Engineering
Isfahan University of Technology, Iran

Uyen Trang Nguyen
Department of Computer Science and Engineering
York University, Canada

1 Introduction

World Wide Web and online social networks (OSNs) have introduced new ways for people to make new friends online and to socialize. OSNs 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 social networks.

The population of potential victims of web-based malware is much larger than that of other types of worms. In addition to the popularity of the World Wide Web, one reason for having larger populations of victims is that web-based worms are not banned through web proxies and network address translation (NAT) processes. Research has shown that web-based malware can propagate much faster than traditional malware (Grossman 2006). As a matter of fact, the first OSN worm that hit MySpace in 2005 by exploiting a cross site scripting vulnerability in a MySpace web application infected about one million victims within 24 hours (Grossman 2006). As Figure 1 shows, this worm, which was named Samy, propagated much faster other traditional worms.

Malware in OSNs can be broadly classified into three types based on the mechanism used to infect users: cross site scripting (XSS), Trojan and clickjacking.

1.1 Cross Site Scripting Malware

Cross site scripting (XSS) is a security flaw that many web applications are vulnerable to (Grossman
The graph in Figure 2a shows the distribution of web application vulnerabilities, among which XSS is the most common threat. While XSS is a common vulnerability in web applications, its threat becomes more noticeable due to the combination of HTML and AJAX technology. AJAX allows a browser to issue HTTP requests on behalf of the user. Thus there is no need for an attacker to trick the user into clicking a malicious link. This technique provides facility for malware writers to create XSS worms.

There are two types of XSS attacks: persistent and non-persistent (OWASP, 2010). In persistent attacks (also known as stored attacks), the injected code is permanently stored on the target servers as HTML text in a database, a comment field, or messages posted on online forums. A victim’s computer then accesses the malicious code on the server when it retrieves the stored information via the browser. Non-persistent attacks (also known as reflective attacks) are the more common type of XSS attacks. In this case, the injected code is sent back to the visitor by the server in an error message, a search result, or any other type of response that reflects some or all of the user’s input in the result (Figure 2b).

![Figure 1: Total number of infections caused by different Internet worms in 20 hours](image)

![Figure 2: (a) Distribution of critical vulnerabilities in web applications. (b) Reflective XSS attack (Grossman 2006)](image)
An XSS worm, also known as a cross site scripting virus, is a malicious code that propagates itself automatically among visitors of a website in an attempt to progressively infect other visitors. A Cross Site Scripting worm is a combination of one of the oldest security issues, i.e. virus with the new age vulnerability in the Web application, i.e. Cross site scripting. XSS worm is a hybrid of stored and reflected XSS attack. This type of worms has the ability to copy itself in other part of Web pages using the existing XSS vulnerability of the Web applications.

XSS worms infect members of a social network in two steps. The worm creator first adds the malicious payload to his profile, e.g., in the form of a link. Subsequently, any person who visits this profile will get infected and the malicious payload will be added to the visitor’s profile due to the XSS flaw and exploiting the AJAX. The visitor’s profile then becomes an infectious profile which allows the worm to propagate as a new infection source (Faghani & Saidi, 2009).

1.2 Trojan Malware

The best known OSN Trojan worm is Koobface (Kaspersky, 2008). This worm was first detected in 2008. It spread itself in both MySpace and Facebook by sending messages with interesting topics using social engineering techniques to deceive people. The message directed the recipients to a third-party website unaffiliated with Facebook where they were prompted to download what is claimed to be an update of the Flash player. If they downloaded and executed the file, they would infect their computers with Koobface. The infected machine turned into a zombie or a botnet. Moreover, the owner of the infected profile unknowingly sent out messages to all people on his/her friend list, allowing the worm to propagate further in the network.

1.3 Clickjacking Malware

Clickjacking worms are also known as “likejacking” or “UI redressing”. A clickjacking attacker creates a website that shows a counterfeit YouTube video player, or other graphical icons, and invites the victim to click on a play button to view the video. What really happens is that the victim is clicking a Facebook "Like" button that has been hidden beneath the images using a method of coding called UI redressing. What the victim has just “Liked” is then displayed on his wall, which in turn may attract his/her friends to click on that link and become new sources of infection (Figure 3).

Clickjacking worms could be combined with Trojan malware to create a new hybrid type but, to the best of our knowledge, no such malware has been created or deployed.

Figure 3: Clickjacking technique used to spread spam messages that may lead to malicious software
The topic of malware propagation in OSNs has recently been investigated in literature (Faghani & Saidi, 2009, Yan et al., 2011, Stein et al., 2011 and Xu et al., 2010). The objective of this chapter is to discuss recent advances on this topic. In particular, we present three malware propagation techniques in OSNs, namely XSS, Trojan and clickjacking types, and their behaviors via analytical models and simulations.

The remainder of this chapter is organized as follows. In section 2, the characteristics of a social network are defined. In section 3, different models of malware propagation in social networks are investigated. Simulations are done in section 4 to offer insights on how the clustered nature of social network can affect malware propagation. Section 5, discusses the related works in OSNs malware research. Finally section 6, concludes this paper.

2 Online Social Networks Characteristics

An OSN can be represented by an equivalent graph in which each vertex (or node) represents a person and a link between two vertices indicates the existence of a relationship between the two respective persons. To simplify the discussions in this chapter, we generalize relationships between OSN users as friendship. (In some OSNs such as LinkedIn, relationships can be colleagues or business contacts). Studies have shown that real-world social networks are highly clustered small-world networks (Watts, 1999) with a degree distribution often following a power law distribution. The characteristics of social networks, which are studied in (Yong-Yeol et al., 2007, Dekker, 2008, Holme & Beom, 2002), can be summarized as follows:

1. A social network typically has a low average network distance, approximately equal to \( \log n/\log d \), where \( n \) is the number of vertices (people), and \( d \) is the average vertex degree of the equivalent graph.

2. 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 a social network graph. Then the clustering coefficient \( C(v) \) of vertex \( v \) is defined as \( C(v) = 2f/(k(k-1)) \). The clustering coefficient of a graph is the average of the clustering coefficients of all of its vertices. In real social networks, the clustering coefficient is about 0.1 to 0.7.

3. Node degrees of a social network graph tend to be, or at least approximately, power-law distributed. The node degree of a power-law topology is a right-skewed distribution with a power-law Complementary Cumulative Density Function (CCDF) of \( F(k) \propto k^{-\alpha} \), which is linear on a logarithmic scale. The power law distribution states that the probability for a node \( v \) to have a degree \( k \) is proportional to \( p(k) \propto k^{-\alpha} \), where \( \alpha \) is the power-law exponent (Newman, 2005).

There exist few algorithms that can generate social network graphs with the above characteristics (Dekker, 2008, Holme & Beom 2002, Davidsen et al., 2002). For the simulations reported in this chapter, we used the algorithm proposed by Holme and Beom (Holme & Beom 2002), because it can be fine tuned to generate a social network graph with the required clustering coefficient and power law distribution of node degrees.

In one of our experiments, we evaluated the speed of malware propagation as a function of cluster-
ing coefficients. For this experiment, we would need to vary the clustering coefficient while keeping other parameters of the network graph such as the maximum and average node degrees constant. To create such similar graphs with different clustering coefficients, we would need random graph generation algorithms such as random rewiring or (Viger & Latapy, 2005). In this section, we discuss the algorithm by Holme and Beom along with these random graph generation algorithms.

2.1 Holme’s Social Network Graph Generation Algorithm

This algorithm is based on the algorithm proposed by Barabasi and Albert (Barabasi & Albert, 1999), which we term BA algorithm. The objective of the BA algorithm is to create graphs with power law degree distributions. These graphs have short average network distances typical of OSNs, but they may not have high clustering coefficients, between 0.1 to 0.7, to faithfully model social network graphs (Holme & Beom 2002). This motivated Holme and Beom to modify the BA algorithm to generate graphs that show clustering coefficients typical of OSNs. The BA algorithm works as follows:

- The initial condition: A graph consists of \( m_0 \) vertices and no edges.
- The growth step: One new vertex \( v \) with \( m \) edges is added to the above graph at every time step. Time \( t \) is identified as the number of time steps.
- The preferential attachment (PA) step: Each of the \( m \) edges incident on \( v \) is then attached to an existing vertex \( u \) with the probability \( P_u \) defined as follows:

\[
P_u = \frac{k_u}{\sum_{i \in V} k_i}
\]

In the above equation, \( k_i \) represents the degree of node \( i \), and \( V \) is the set of vertices of the current graph. The growth step is iterated \( N \) times. Every time a vertex \( v \) with \( m \) edges is added to the network, the PA step is performed \( m \) times, once for each of the \( m \) edges incident on \( v \). After \( t \) time steps, the BA network graph will contain \( m_0 + t \) vertices.

To increase the clustering coefficient, Holme and Beom suggested a new step called triad formation (TF). If in a PA step an edge between \( u \) and \( v \) is formed, then a TF step will attempt to add another edge between \( v \) and an arbitrary neighbor \( w \) of \( u \). If all neighbors of \( u \) have already been connected to \( v \), the TF step is skipped and a new PA step will start to add a new vertex to the graph.

In each iteration, a PA step is first performed: a vertex \( v \) with \( m \) edges is added to the existing network. Then a TF step is executed with probability \( P_t \). The average number of the TF trials per added vertex is given by \( m_t = (m-1) P_t \), which is a control parameter in Holme’s model. It has been shown that the degree distribution of the graph follows a power law distribution with \( \alpha = 3 \).

We used Holme’s algorithm to generate a graph that has the characteristics of a social network and the following parameters: \( \alpha = 3 \), \( n = 10000 \), \( m = m_0 = 3 \) and \( m_t = 1.8 \). The parameters of the resulting social network graph are listed in Table 1.

As Table 1 shows, the generated social network graph satisfies all the three required characteristics of an OSN. The average shortest path length of the graph is less than \( \log n / \log d \). The clustering coefficient is moderate, approximately 0.14. The degrees of the vertices follow a power law distribution, as shown in Figure 4.
2.2 Random Graph Generation Algorithms

To evaluate the impacts of clustering coefficients on worm propagation, we would need to keep the other parameters of a graph such as the maximum and average node degrees constant while varying the clustering coefficient. Such graphs with different clustering coefficients can be generated as similar random graphs. Given a network graph, a similar random graph with the same vertex degree distribution can be generated using random rewiring as shown in Figure 5 or the algorithm defined by (Viger & Latapy, 2005).

In the random rewiring scheme, we randomly select a pair of edges and “substitute” the edges as shown in Figure 5. The random selection and substitution are done until we obtain the desired clustering coefficient.

<table>
<thead>
<tr>
<th>Graph Characteristic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of vertices (people)</td>
<td>10000</td>
</tr>
<tr>
<td>Number of edges (friendships)</td>
<td>29990</td>
</tr>
<tr>
<td>Average clustering coefficient</td>
<td>0.1409392</td>
</tr>
<tr>
<td>Average shortest path length</td>
<td>5.133096</td>
</tr>
<tr>
<td>Maximum node degree</td>
<td>190</td>
</tr>
<tr>
<td>Longest path length (network diameter)</td>
<td>10</td>
</tr>
<tr>
<td>Average node degree (d)</td>
<td>5.998</td>
</tr>
<tr>
<td>logn/logd</td>
<td>5.1413</td>
</tr>
</tbody>
</table>

Table 1: Parameters of the social network graph we generated using Holme’s algorithm

Figure 4: The degrees of the vertices of the resulting graph follow a power law degree distribution.
Figure 5: Random rewiring technique to generate similar random graphs

The algorithm by (Viger & Latapy, 2005) can also be used to generate a random graph with a given degree sequence of vertices. This random graph has the same degree distribution as the original network graph, but different characteristics such as a different clustering coefficient, average shortest path length or network diameter.

The random rewiring algorithm generates random graphs that have more correlation to the original graph than the algorithm by Viger and Latapy. We would want random graphs that are as versatile as possible while maintaining the same vertex degree distribution of the original graph. Therefore, we chose the algorithm by Viger and Latapy to generate similar random graphs for our simulations. The parameters of one such similar random graph are listed in Table 2. The original graph used to compute this similar random graph is the one described in Table 1.

<table>
<thead>
<tr>
<th>Graph Characteristic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of vertices (people)</td>
<td>10000</td>
</tr>
<tr>
<td>Number of edges (friendships)</td>
<td>29990</td>
</tr>
<tr>
<td>Average clustering coefficient</td>
<td>0.003581474</td>
</tr>
<tr>
<td>Average shortest path length</td>
<td>4.407071</td>
</tr>
<tr>
<td>Maximum node degree</td>
<td>190</td>
</tr>
<tr>
<td>Longest path length (network diameter)</td>
<td>8</td>
</tr>
<tr>
<td>Average node degree ((d))</td>
<td>5.998</td>
</tr>
<tr>
<td>(\log n/\log d)</td>
<td>5.1413</td>
</tr>
</tbody>
</table>

Table 2: Social Network Graph Characteristic

Following are some observations obtained from comparing the similar random graph (Table 2) and the corresponding original graph (Table 1). The average clustering coefficient of the original social network graph is about 40 times higher than that of the similar random graph. This reflects the high clustering characteristic of social networks. Also, the average shortest path length of the original graph is longer than that of the random graph, 5,133 vs. 4.407. These differences are due to the small world phenomenon described in (Watts, 1999).
3 Modeling Worm Propagation in Online Social Networks

OSN worms, like other computer worms, behave in a similar manner to biological viruses in terms of infectious disease propagation. Therefore, mathematical analyses on propagation behaviors of biological viruses can be adapted to studies of computer worms (Bulgin Y., 2007, Zou, et al., 2002).

In the area of epidemiology, infectious disease propagation can be modeled using either stochastic or deterministic models (Andersson & Britton, 2000). Stochastic models are suitable for a small scale population, while deterministic models are for a large scale population. Deterministic models should thus be used for modeling OSN worm propagation because of large sizes of OSNs. (As of December 2011, Facebook has approximately 800 million users around the world.)

One of the most popular differential equation models for biological worm propagation is the susceptible infected (SI) model. This model has used in several computer worm propagation models such as (Faghani & Saidi, 2009), (Zou et al., 2006), and (Staniford et al., 2002). In this section, we discuss the SI model and existing models proposed for OSN worms.

3.1 The SI Model

The SI model is defined as follows:

\[
\frac{dI(t)}{dt} = \frac{\eta}{\Omega} I(t) [N - I(t)]
\]  

(2)

In this model, N is total number of people in the population; I(t) is the number of infected hosts at time t; \( \eta \) is the worm infectious activity rate; and \( \Omega \) is the number of possible targets that can be reached by the worm. All hosts are assumed to be either vulnerable (susceptible) or infected according to the SI model. In the field of epidemiology, susceptible hosts are defined as those vulnerable to infection by the virus. Infectious hosts are those that have been infected and can infect others. A host is considered infected at time \( t \) if it had been infected before time \( t \).

Assuming that \( \eta \) is not a time variant variable and the initial condition is \( I(t) = i_0 \), the solution to equation (2) is as follows:

\[
I(t) = \frac{i_0 N}{i_0 + (N - i_0)e^{-\eta \Omega t}}
\]  

(3)

The SI model has been used to model cross site scripting worms (Faghani & Saidi, 2009) as well as Trojan and clickjacking worms in OSNs.

3.2 Modeling Cross Site Scripting (XSS) Worms

In (Faghani & Saidi, 2009), the authors suggested the following model for cross site scripting worms:

\[
\frac{dI(t)}{dt} = \beta(q) \frac{I(t)}{N} (N - I(t))^{K(q)}
\]  

(4)

\( \beta(q) \) is the rate of visiting friends in an OSN, which is a function of the friend visiting probability \( q \). The friends of a user are defined as those whose nodes are adjacent and connected to the user’s node in the network graph. \( q \) is the user’s probability of visiting friends as opposed to strangers (those whose nodes are not connected to the user’s node in the network graph). \( K(q) \) represents the sensitivity of the susceptible population to \( q \). In general, \( K(q) \) is proportional to \( q \) (Faghani & Saidi, 2009). As \( q \) increases, members will visit their friends more often than strangers. If the number of infected friends is small, a large
value of $q$ will delay the propagation of worm in their community. Therefore, uninfected users are also sensitive to (affected by) the value of $q$, which $K(q)$ takes into account.

The model represented by Eq. in (4) is based on the following two facts. First, when the friend visiting probability $q$ increases, the infection is more contained among friends. Less strangers would be affected by infected users. Therefore, the delay for the infection to reach other parts of the network will be longer. In other words, the rate of infection is inversely proportional to the friend visiting probability $q$. Second, after most of the users are being infected, the infection rate will slow down since the total number of susceptible hosts has also been decreased.

### 3.3 Modeling Trojan and Clickjacking Worms

Let $P(k)$ be the probability that a node in the network graph has degree $k$. The average degree of the network is thus $E[k] = \sum_k kP(k)$. Suppose that the fraction of infected users having degree $k$ is $i_k(t)$. Let $\lambda$ be the infection rate, which is the probability of getting infected by an infectious neighbor in a time unit. The differential equation model for nodes with degree $k$ is given by the following (Moreno et al., 2003):

$$\frac{di_k(t)}{dt} = \lambda k [1 - i_k(t)] \Theta(t)$$

$$\Theta(t) = \frac{\sum_n nP(n)i_n(t)}{\sum_n nP(n)} = \frac{\sum_n nP(n)i_n(t)}{E[k]}$$

The factor $\Theta(t)$ is the probability of a link pointing to an infectious profile. That is, the probability of a link pointing to a $k$-degree node is $kP(k)$ (Moreno et al., 2002)

The above model is later improved in another model proposed by (Boguna, et al., 2003), which considers the fact that the originator of an infection will not be infected again (by its child in the spanning tree of the network graph). The revised model is as follows:

$$\frac{di_k(t)}{dt} = \lambda k [1 - i_k(t)] \Theta(t)$$

$$\Theta(t) = \frac{\sum_n (n - 1)P(n)i_n(t)}{\sum_n nP(n)}$$

The total number of infected nodes $I(t)$ would be:

$$I(t) = \sum_k i_k(t) P(k)N$$

The model defined by Eq. (6) and (7) above, however, is not accurate for modeling Trojan and clickjacking worms in OSNs because it does not consider important factors such as the clustering coefficient or the relationships and activities (communications) among infected nodes. This is still an open issue for future research on modeling malware propagation in OSNs.

### 4 Simulation of Malware Propagation in Online Social Networks

In this section we present our simulation results on malware propagation in OSNs. The simulation soft-
ware is implemented using MATLAB. The simulation is of discrete-event type, consisting of discrete time slots. In each time slot, a user (node) is chosen randomly and the user will perform an action such as visiting a friend’s (or stranger’s) profile, or executing the malware.

In all the simulations discussed below, we used the simulated social network created using the Holme’s algorithm whose parameters are listed in Table 1, unless otherwise stated.

The performance metric is the total number of infected profiles (users) over time, assuming an initial number of infected profiles of one, unless otherwise stated. Each data point in the result graphs is the average of 100 runs, each with a different random seed.

We measure the total number of infected profiles over time as functions of the following varying parameters:

- **Friend visiting probability** \( q \). The probability that a user visits his/her friends versus strangers. A friendship exists between two users \( u \) and \( v \) if there is an edge connecting nodes \( u \) and \( v \) in the network graph.

- **Graph structure**. A graph can be an *original social network graph* created using Holme’s algorithms, or a *similar random graph* generated based on an original social network graph using the algorithm by (Viger & Latapy, 2005).

- **Probability** \( p \) **of executing the malware by a user**. For XSS worms, this is the probability of having an add-on protection (e.g., no-script add-on for Firefox) active while browsing. For Trojan or clickjacking worms, this is the user’s probability of following a malicious link.

- **Node degree threshold for visiting friends vs. strangers**. Let \( M \) be the maximum node degree in the network, and \( K_c \) be a threshold factor \( (0 < K_c < 1) \). Nodes with degrees less (more) than the threshold \( K_c M \) will visit their friends less (more) frequently than strangers. That is, users whose numbers of friends (node degrees) are lower than the threshold \( K_c M \) will visit their friends with probability \( q < 0.5 \) and visit strangers with probability \( 1 - q \).

- **Initial number of infected profiles (users)**.

We first present the simulation results on XSS worm propagation, followed by the results on Trojan and clickjacking worms. As mentioned earlier, Trojan and clickjacking worms spread in a similar fashion since both send the malware globally to all friends of a user.

### 4.1 Simulation of XSS Worm Propagation

In order for an XSS worm to propagate, a vulnerable user has to visit an infectious profile (user) to get infected. The user’s vulnerability is determined by whether the user’s web browser is able to execute the malicious script or not. (A browser may not be able to execute the script because the user has disabled Javascript using special add-ons such as NoScript for a Firefox browser.)

To simulate the trend of XSS worm propagation, the event is defined as a single visit on the social network website. Therefore, if the visitor is vulnerable and visits an infected profile, then the visitor gets infected.

#### 4.1.1 Effects of Friend Visiting Probability \( q \)

In this experiment, a user visits one of his/her friends with probability \( q \), or picks randomly a stranger to
visit with probability of $1-q$. In each time unit, a user is chosen randomly using a uniform distribution for visiting a friend. We assume that all users have the same friend visiting probability $q$. Figure 6 shows the trend of XSS worm propagation for different values of $q$. The simulation results show that if people visit their friends more often, the propagation speed will be slower. This confirms the analytical model proposed by (Faghani & Saidi, 2009) as described in Section 3.2.

Figure 6: Trend of malware propagation using simulation for different values of $q$

4.1.2 Effects of the Network Graph Structure

![Graph showing trend of XSS worm propagation for different graph structures]

Figure 7: Trend of XSS worm propagation for different graph structure with $q=0.9$

Although the model by (Faghani & Saidi, 2009) takes into account the friend visiting probability, it does not consider the impacts of the clustering coefficient on XSS worm propagation. In this experiment, we
used the social network graph and a similar random graph whose parameters are listed in Table 1 and Tables 2, respectively. We assumed a friend visiting probability $q = 0.9$ on both network graphs. Figure 7 shows the trends of XSS worm propagation for both graphs. Although both networks share the same friend visiting probability and other parameters (e.g., maximum and average node degrees), their results are different. The propagation was slower in the original network graph (i.e., the small-world graph) thanks to its higher clustering coefficient. A user in the small-world graph, on average, has more friends than the same user in the similar random graph due to the higher clustering coefficient. This makes the user visit more friends than strangers, slowing down the XSS worm propagation.

We repeated the above experiment, but changed the friend visiting probability to $q=0.1$. The results in Figure 8 show that both networks experienced the same XSS worm propagation speed in this case.

![Figure 8: Trend of malware propagation for different graph structure with $q=0.1$](image)

The above results show that although both graphs have the same degree distribution, they reflect different worm propagation behaviors with high visiting friend probabilities. This is one of the facts that the current analytical models do not consider. The graph topology is meaningful only for higher probabilities of visiting friends. A low friend visiting probability means that people randomly select someone to visit, independent of the graph topology, which helps the worm to propagate faster.

Furthermore, one of the differences between the social network graph and its corresponding random graph is that the social network graph is highly clustered. This feature slows down the worm propagation given higher friend visiting probabilities.

### 4.1.3 Effects of the Probability $p$ of Executing the Malware by a User

Another important parameter in malware propagation is the probability of executing the malware. That is, if a user receives a malicious link or a file, what is the probability of following the link and executing the code contained in the file? From the XSS type perspective, since the user may get infected unknowingly, what is the probability of having an add-on protection (e.g., no-script add-on for Firefox) active while browsing? Let $p$ denote this probability. Figure 9 shows the trend of propagation over time for different distributions of $p$ and $q$, the friend visiting probability. The results show that the impact of visiting friends as opposed to strangers is dominant in malware propagation.
Figure 9: Trend of malware propagation for different distribution of $p$ and $q = 0.1, 0.9$

4.1.4 Effects of Node Degree Threshold for Visiting Friends vs. Strangers

Initially when users join an online social network, they start looking for friends by visiting different profiles. Hence, they visit strangers more frequently than their friends. In this simulation, we assume that a fraction of people with low numbers of connections will visit strangers more frequently than their friends. Let $M$ be the maximum node degree in the network, and $K_c$ be a threshold factor ($0 < K_c < 1$). Nodes with degrees less (more) than the threshold $K_c M$ will visit their friends less (more) frequently than strangers. In this experiment, users whose numbers of friends (node degrees) are lower than the threshold $K_c M$ will visit their friends with probability $q$ where $q$ has a normal distribution with $\mu = 0.25$ and $\sigma = 0.05$. The other users (i.e., those whose node degrees are higher than the threshold $K_c M$) will visit their friends with probability $q$ where $q$ has a normal distribution with $\mu = 0.75$ and $\sigma = 0.05$. The trends of worm propagation for are $K_c = 0.1$ and $K_c = 0.5$ (equivalent to a threshold of 19 friends and 95 friends, respectively, given the network in Table 1 which has a maximum node degree of 190) are shown in Figure 10.

Figure 10: Different visiting distribution for online social networks
The results show that the propagation is slower when $K_c = 0.1$. The reason is that when $K_c = 0.1$ there are 281 users that have more than 19 friends in contrast to 12 users having more than 95 friends when $K_c = 0.5$. When $K_c = 0.1$, more users visit their friends more frequently than when $K_c = 0.5$, which leads to slower worm propagation.

4.2 Simulation of Trojan and Clickjacking Malware Propagation

In this simulation, we assume that in each time unit, a user checks his/her messages on the OSN. With probability $p$, the user follows a link in the message and executes the malware. Once the user gets infected, he sends a similar message containing the malicious link to all of his friends (or posts it to his wall as in Facebook clickjacking type).

4.2.1 Effects of the Probability $p$ of Executing the Malware by a User

The trend of Trojan malware propagation is shown in Figure 11 for different probability distributions of $p$. We can see that the higher the probability of executing the malware, the faster the worm propagates. Given $p$ with a uniform distribution having $\mu = 0$ and $\sigma = 1$, the propagation is very slow because people are unlikely to open the messages and follow the malicious links.

![Figure 11: Trend of malware propagation for different probability distribution of $p$](image)

4.2.2 Trojan vs. XSS Propagation

In this simulation, we compare the propagation speed of Trojan and XSS worms under the same network conditions and parameters. We assume that people visit other users (profiles) following a Poisson process with an average of $k$ times per minute. Thus the interval between visits follows an exponential distribution with an average of $1/k$. Given $k = 10$ and a malware executing probability $p$ where $p$ has a normal distribution with $\mu = 0.5$ and $\sigma = 0.7$, Figure 12 shows the results of propagation speeds of Trojan and XSS worms in OSNs. The results demonstrate that the propagation speed of Trojan type malware is faster than that of XSS worms in OSNs.
To explain the above result intuitively, we consider a tiny social network depicted in Figure 13. Initially user $A$ is infected while the others are not. If user $A$ is infected with a Trojan malware and she has already sent the malicious message to all her friends, $B$, $C$, $D$ and $E$, then in the next visit (event) an uninfected user will be selected with a probability of $4/5$. Assuming that this user will open the message and follow the malicious link, this means that one of the uninfected users will get infected with probability of $4/5$.

However, if user $A$ is infected with an XSS worm, then in the next visit, an uninfected user will be selected with a probability of $4/5$ and this user will visit the infected user $A$ with a probability of $1/4$. Therefore, one of the uninfected users will get infected with probability of $4/5 \times 1/4 = 1/5$. This explains why the Trojan worm propagated much faster than the XSS worm in the above simulation.

In other words, Trojan worms are more proactive than XSS worms. They present themselves to users (in the form of messages) so as to be activated and propagated, while an XSS worm sits on an infected profile waiting for users to select and visit the profile.

**Figure 12:** Trend comparison over time for XSS and Trojan type malware

**Figure 13:** A tiny social network in which user $A$ is initially infected while the others are not
4.2.3 Effects of the Initial Number of Infected Profiles

Another important parameter that should be considered in the propagation speed of Trojan and XSS worms is the initial number of infected profiles (users) \( i_0 \). In (Faghani & Saidi, 2009), it is proved that the initial number of infectious profile affects the speed of propagation logarithmically for both Trojan and XSS worms. Assuming the network graph in Table 1, we varied the initial number of infected profiles from 50 to 500, and measured the number of visits (events) required to get 90% of the population (i.e., 9000 users) infected. The obtained results were then normalized to the maximum number of visits measured for each type of worm. Figure 14 shows the results of this experiment. We can see that the effect of increasing the initial number of infected profiles for a Trojan worm such as Koobface is not noticeable. However, increasing the initial number of infected profiles for XSS leads to faster propagation.

![Figure 14: XSS versus Trojan type for different values of initial infected profiles](image)

In Trojan malware propagation, assume that each person follows the malicious link and executes the malware code with a probability of 1. Suppose that there are initially \( n \) infected profiles. Thus, on average, there are \( n \times \text{deg}(n) \) potential targets for infection, where \( \text{deg}(n) \) is the average degree of the \( n \) infected nodes.

Therefore by increasing the initial number of profiles infected by a Trojan malware \( i_0 \), it is possible to make almost all members become potential targets for infection. If we keep increasing \( i_0 \), the increase may not have significant effects on the propagation speed.

We now explain why the initial number of infected profiles affects the speed of propagation logarithmically. In the SI model, the time required to infect \( k \) percent of the vulnerable population, or \( kN \) people, is logarithmically proportional to the initial number of infected people \( i_0 \), as proved below. From Eq. (3) in Section 3, we obtain:

\[
\frac{i_0 N}{i_0 + (N - i_0) e^{-\eta N t}} \geq kN
\]

\[
t \geq \ln \left( \frac{N - i_0}{i_0} \right)^{\frac{\Omega}{\eta N}}
\]  

(8)
Our simulation results confirm the above model: increasing the initial number of infections will decrease the time required to infect 90% of the vulnerable population logarithmically.

5 Malware Propagation Countermeasures

Several malware detection and containment mechanisms for OSNs have been proposed recently. In (Nguyen et al., 2010), the authors proposed a centralized patch distribution algorithm which monitors the number of infected users. When the fraction of infected users reaches a pre-determined threshold, the detection system raises the alarm and sends out “treatment” patches to influential users. Influential users of a community are those having large numbers of relationships (connections) with other communities. They are thus the best candidates to distribute the “treatment” patches efficiently throughout the whole network. After receiving “treatment” patches, a user will apply them to eliminate the worm and forward them to his/her friends.

In (Xu et al., 2010), the authors suggested a scheme in which by monitoring a small fraction of users, the entire network can be under surveillance. An early detection will allow for effective worm containment and elimination measures.

Stein and his colleagues from Facebook described the Facebook immune system in (Stein et al., 2011). Their immune system performs real-time checks on every incoming and outgoing query to network. To defend against malware, their classifier identifies infected users when they send many messages flagged by the classifier or other users.

Yan et al. also investigated the issues of malware propagation in OSNs (Yan et al., 2011). They introduced two defense strategies against malware: one is user-oriented defense and the other, server-oriented defense. In the user-oriented defense, after an infected user recovers, he/she sends a warning message to his/her friends, who may in turn forward the warning message to their friends. In the service-oriented defense, the server selects a number of users and checks messages sent and received by them to make sure that no malicious links are included in the messages. The immune system used by Facebook is a form of service-oriented defence.

The models and simulation results presented in Sections 3 and 4 suggest that one of the most efficient ways to defend against malware in OSNs is to detect it early within communities. The reason is that a malware will circulate among members of a community for a while before it gets a chance to move to another community, due to the high clustering property of OSNs. This approach of malware detection and containment is still an open issue for future research.

Another optimized defending mechanism is to take into account portions of the network graph which are built based on active relationships among OSN users, since active users are more likely to visit friends and execute malware code. In (Wilson et al., 2009), it is shown that the active graph of user interactions is different from the actual graph of relationship in an OSN.

6 Related Work

There exists research in the field of epidemiology that models the behavior of contagious diseases (Moreno, et al., 2003, Boguna, et al., 2003, Pastor-Satorras & Vespignani, 2001).

Malware propagation in general social networks has been studied extensively recently. Most of them focus on epidemic spreading analyses. Malware propagation in small-world networks has been studied in (Telo & Nunes, 2006, Moore & Newman, 2000). Malware propagation in scale-free network structures has been investigated in (Moreno et al. 2003, Griffin & Brooks, 2006) Malware propagation in spe-
cific types of networks such as e-mail, instant messaging (IM) and mobile networks has also been well studied (Mannan & Oorschot, 2005, Zou et al. 2005, Cheng et al., 2011).

Among the first works targeted malware propagation in OSNs are those by (Faghani & Saidi, 2009), (Yan et al., 2011), and (Xu et al., 2010). Faghani and Saidi modeled the propagation of XSS worms using the SI model, and investigated malware propagation in OSNs using simulated topologies and based on user activities. (Yan et al., 2011) used realistic network topologies in addition to realistic user activities to confirm that user activity models play an important role in malware propagation in OSNs. (Xu et al., 2010) proposed a correlation-based scheme to mitigate worm propagation in OSNs. Their work was based on Flickr network.

There exists also research on human activities in OSNs. Benevenuto et al. analyzed user activities on four popular OSNs (Orkut, Hi5, Myspace and LinkedIn) and provided useful information on how users behave in OSNs (Benevenuto et al. 2009).

7 Chapter Summary

We discuss the behaviors of malware propagation in online social networks using analytical models and simulation results. In general, the propagation of XSS worms depends largely on users’ behaviors: If OSN users visit mostly their friends rather than strangers, the worms will propagate more slowly. The highly clustered feature of social networks also helps to slow down the propagation. Increasing the initial number of infected profiles in the early stages of XSS worm propagation leads to an impressively faster propagation. Trojan worms propagate faster than XSS worms in social networks because of their inherent aggressive propagation method. Increasing the initial number of infected profiles in the early stages of Trojan worm propagation does not have considerable effects on the propagation speed.

We also identify open issues for future research. First, current analytical models do not consider the network graph structure in the propagation of malware in OSNs. Second, we should exploit the high clustering structure of OSNs to detect the propagation of XSS worms early within a community, e.g., using a honeypot detection mechanism.

References


