ABSTRACT

Social networks have boomed over the past few years. On a daily basis, countless users send messages and post or share information over the internet on these networks. Therefore it is no surprise that online social networks are a widely used medium to spread malicious software. Numerous researchers have tried to model the propagation of social network worms. For some of these networks is malware propagation not unheard of. In this paper a worm propagation model is developed that describes the propagation through social networks which is, among other things, not limited to decentralization or encryption characteristics. Infection simulations are run with three different infection probabilities as well as three different sets of initial seed vertices. Different containment methods are reviewed and, if possible, run on a dataset, which sampled a portion of the Facebook network, to verify how sufficient they are and to review at which stages the methods are the most effective.

Keywords
Worm propagation, Social network, Facebook, Independent Cascade Model, Worm containment

1. INTRODUCTION

Facebook and WhatsApp are the two largest social- and instant messaging networks available with their monthly active users being respectively 1.86 [18] and 1.2 billion [19]. Hackers abuse the fact that users perceive messages they receive from their friends as trustworthy. "Human error is often a major cause of problems in technological implementations, and people are generally considered the weakest link in an information security program" [1]. Therefore these platforms make for a very attractive medium to spread all kinds of malicious software (malware). Social network worms are a type of virus, able to spread without requiring to attach itself to a host file. This means the virus itself contains all the necessary code to propagate through the network which makes it optimal for social networks. In the past the Facebook network was used successfully to distribute self propagating malware and infect its users. [17]. WhatsApp is no exception and has also been targeted by hackers for their personal gain.

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

27th Twente Student Conference on IT, July 7th, 2017, Enschede, The Netherlands.
Copyright 2017, University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science.

A scam called 'Facebook Gold' tricked users into downloading a supposedly premium version of the WhatsApp app [3]. This app then had access to the contacts list and would try and trick those users as well. Some networks have applied multiple measures to prevent these kinds of infections. Facebook has a custom centralized datacenter. The data that goes through these servers is not encrypted, which means that they have the ability to check what is being sent. They also have a software system called Sigma, which is a ruling engine for posts, messages, status updates, etc [13]. Sigma checks the data on several policies and determines whether the data is clean or contains some kind of malicious links, programs or data and filters accordingly. Unfortunately not every social network or instant messaging app has the luxury of their own centralized datacenter and sophisticated anti-malware software. Jabber (now known as XMPP) is a protocol that messaging applications can implement [24]. This protocol is widely used by multiple standalone apps. Examples are Adium, Pidgin, ChatSecure, etc [23]. Users can chat with other people that use a messaging app, if it follows this protocol. The difference between this network and, for example, Facebook can be described by the underlying architecture of the network. The XMPP network is built out of individual servers that anyone can run. Users can connect to this server and are then able to send messages to other users in the network. Combined, these individual servers function as a decentralized data center. Decentralization, however, makes controlling what data is being sent through the network harder because it does not have a single point of entry. An added problem, for malware prevention, is that some applications apply end-to-end (e2e) encryption. When e2e encryption is applied, the data is only visible to the end users and ensures no-one has the ability to intercept the actual messages. This means that you cannot apply some rules and policies to the data sent through the server. An example is WhatsApp. Because of the encryption their application is possibly less secure than others that do not apply this encryption. These applications have been targeted with malware in the past as well. Jabberbot was a command and control malware which executed malicious code on the users' computer [6]. Even though this was a minor incident, the principle used for the attack could have had major effects on the users of the network. Due to a poor design the malware was not as efficient as it could have been [6]. These accidents show that malware propagating through these kinds of networks is very realistic and research into these topics is therefore very important.

2. BACKGROUND
Large social networks often have a scale-free degree distribution (which follow a power law) [11]. Among the models that describe information flow through such networks, there are a few models originating from the Cascade family. These models have been widely studied previously for influence maximization problems mainly for marketing purposes [7][8] but have a wide range of applicability. The Independent Cascade (IC) model is a very popular variant. The following definition is obtained from [16]. The model assumes one or more initial vertices in a network that adopt the technology (in this case malware). These vertices have a single possibility to infect each subsequent vertex they have access to and will successfully infected them with a certain probability. This probability can be independent for each individual edge. These infection attempts are described in discrete time steps.

If infection is successful the newly infected vertex can attempt to infect each of the vertices it is directly connected to in the next time step. The previously infected vertices become passive and have no further possibilities to infect others. This process repeats itself until no further infections are possible either due to everyone being infected, or by termination of the propagation because no more active vertices exist.

A single possibility for infection might be counter-intuitive, but is actually very realistic for social networks, when considering social engineering. Psychological engineering is the term for using deception to manipulate users into providing confidential or personal information which is then used for malicious purposes [22]. Using human emotions like fear, curiosity, greed or excitement are often key to obtaining information [1]. An experiment [9] showed using panic that people easily let down their guard when over-flooded with emotions. Incorporated in the case of social network worms, this could result in links being personalized to a user’s interest which could dramatically increase the chance of infection. The message or post that a user receives therefore is, in theory, a realistic attempt. An added benefit is that these users have a relationship with each other and therefore there is a basis of trust as well as a certain expectation. If this attempt does not succeed there is no reason why any further attempt at infection would succeed.

For clarification, the process of a cascade infection propagation[16] has been visualized in Figure 1. Green heads represent healthy but susceptible vertices. Red heads represent infected vertices that are actively trying to infect its neighbors. Grey heads are infected vertices that have previously attempted to infect their neighbors but are now no longer actively infecting others. Finally the red dashed edges show the infection flow for that time step. There are different cascade models developed, but the IC distinguishes itself with its probability of infection. The IC model treats every edge independently and assigns a probability, which can be anything. For example it can be based on a certain property specific for the edge between two vertices (e.g. path length, amount of interaction, friendship status or otherwise) but can also be completely arbitrary.

3. RESEARCH QUESTIONS

For this research the following questions are formed:

RQ1 What is a realistic worm propagation model for social networks that could have, among other characteristics, decentralized data centers or end-to-end encryption?

RQ2 How well do current worm-containment methods perform with the aforementioned propagation model?

4. RELATED WORK

Prior research has given us some insight into social network malware propagation. [20] investigates scanning based worms and worms that have topological awareness. Such worms scan through an IP-address space and look for exploitable hosts and have been extremely successful in infecting large amounts of systems within a short amount of time. Although this kind of research might be essential for developing propagation models [20], it does not present a concrete model for current social networks. A study performed in 2013 [21] has lead to a propagation model that has potential for power-law based networks and among other things factors in the time at which the user opened a malicious link. The idea was that other models’ estimation of infection probability was overestimated due to the fact that upon sending an infection attempt, the receiving vertices instantly become infected and start infecting others.

Their research factors in a certain delay, but with the evolution of social media and smartphones in current day and age this statement can be questioned because a study [2] performed by the Nottingham Trent University found that people check their smartphones for an average of 85 times a day and spend approximately 5 hours — one third of the time they are awake — each day browsing the web and using mobile applications. Therefore any message sent will
probably be opened in a relatively instant matter.
In essence email worm propagation is very similar to in- 
stant messaging and social network worm propagation and 
is therefore also of interest for this research. Worms spread-
ing via email search through address lists and send those 
users messages which are meant to trick them into open-
ing an attachment. This behavior is very similar in these 
social network platforms where an attachment is substi-
tuted for a link or an executable. [25] looks at a process 
similar to IC worm propagation in emails. One problem is 
that their research is relatively old and the probability for 
infection is very high (50% infection probability). If such 
attacks would take place nowadays the probability for 
success would be much lower due to, among other things, 
more experience and general knowledge about these kinds 
of attacks. This has created more awareness and lowered 
its effectiveness.
[14] references an epidemiology model, the susceptible-
infected-removed (SIR) model. The SIR model does not 
describe the propagation of worms but describes the states 
that vertices in a network can be in. This model is the pre-
cursor of the IC model, with the same vertex states. In 
their research they simulate two different types of graphs;
graphs based on the Watts-Strogatz model (which has 
small-world properties) and Barabasi-Albert model (which 
has power-law and scale-free properties). These are both 
relevant because social networks show similar network prop-
erties.

Worm containment research comes in multiple shapes and 
sizes. We distinguish two types. One is more graph the-
ory oriented and looks at blocking links between clusters 
[15]. Others focus on influence maximization to distribute 
patches to the users with the most influence for a high 
impact. The second type of methods assume further dis-
tribution is done manually by the users of the network and 
therefore a high impact from initial users is beneficial. 
However, this containment method is very unlikely to be 
used in the real world, at least for networks similar to 
WhatsApp or with the XMPP protocol. The problem 
with this method lies in the essence of its use, because 
patch distribution by hand is rarely used anymore. This 
is especially the case when everyone uses a downloaded 
client (which is updatable). If we take WhatsApp as an 
example, whenever there is a patch (whether this is be-
cause there is a regular update or an exploit in the app is 
being abused) they push a new version to the designated 
places (e.g. the Android store). The users can then install 
the update through their phones’ regular update system. 
This contradicts with the idea of distributing patches to a 
set amount of users that supposedly have the most influ-
ence, which would subsequently spread this patch to their 
friends manually.

5. METHOD OF RESEARCH

5.1 Infection algorithm
In this paper we propose a malware propagation model 
based on the IC model presented in [16]. The simplicit-
ity and solid foundation from prior research makes this 
model suitable for modeling malware propagation. This 
model takes into account the propagation of a social net-
work worm which is not restricted to single data centers 
nor relies on an encryption status. The algorithm has 
three distinct parameters. The first is a graph of the net-
work $G = V, E$ consisting of vertices $V$ and edges $E$. It 
has to be undirected. The second is a set $S$ of vertices 
$\in V$ that represent the initially infected users and finally

\begin{algorithm}
\caption{Propagation of a social network worm. $G$ is the graph, $S$ the set of seed vertices, $p$ probability of infection.}
\begin{algorithmic}
  \Procedure{Propagate}{$G, S, p$}
  \State $H \leftarrow \{G - S\}$
  \State $A \leftarrow S$
  \State $\text{temp}_A \leftarrow \{\}$
  \State $P \leftarrow \{\}$
  \While{$H \neq \{\} \land S \neq \{\}$}
  \State \ForEach{$n \in A$}
  \State \ForEach{$k \in (G, \text{get_neighbors}(n) \cap H)$}
  \State $ip \leftarrow$ random number between $(0,1)$
  \State \If{$ip < p$}
  \State $\text{temp}_A \leftarrow \text{temp}_A + k$
  \State Remove $k$ from $H$
  \EndIf
  \EndFor
  \EndFor
  \State $P \leftarrow P + A$
  \State $A \leftarrow \text{temp}_A$
  \EndWhile
  \State \Return $P$
  \EndProcedure
\end{algorithmic}
\end{algorithm}

an infection probability $p$, which can be anything rang-
ing from 0% to 100% and is chosen manually in our case. 
Algorithm 1 is a pseudo code representation of the propaga-
tion algorithm.

When talking about worm propagation there are at least 
two types, one is primarily based on tricking users through 
social engineering scams where the users are the flaw in 
the system, whereas other worms abuse an exploit found 
in the system. In the case of social engineering the infec-
tion probability usually is much lower compared to when 
an exploit in a system is being used due to the complex-
ity of a good social engineering scam. For this reason, we 
chose three different infection percentages which reflect a 
couple different situations that could happen in the real 
world.

We have chosen to run simulations on three different infec-
tion percentages $p$. These percentages were chosen after 
consulting with a security researcher from the University 
of Twente. The first choice, 1% represents the least ef-
ective attack, a social engineering attack. The second 
percentage, 5% represents a more advanced attack which 
could incorporate a small exploit and the final option, 10% 
represents a large exploit which is easy to abuse and there-
fore has the most effect. A very recent example of such an 
attack was WannaCry which used an SMB exploit in Wind-
ows called "EternalBlue" [10]. We need to get an accurate 
estimation of how many vertices will be infected for a cer-
tain percentage $p$. This value is required to be able to say 
anything about the upcoming containment simulations.

For the simulations an undirected graph, which is a sam-
ple of the Facebook network, is used [12]. This graph has 
a total of 4039 vertices and 88234 edges and is one giant 
component which is strongly connected. It has an average 
clustering coefficient of 0.6055 and contains 8 large clus-
tered regions which are connected through bridge vertices. 
Besides the infection percentage another important aspect 
is the set of seed vertices. Within a network there are ver-
tices that each have a different role. For example there 
are bridge vertices (these connect different clusters with 
each other) while other vertices have an authority role in 
the network with lots of ingoing links and others are bet-
ter described as hubs. For this reason, three sets of seed
vertices have been chosen which differ either in placement or size and therefore also in outcome. One set, represented in Figure 2(a) (from now on denoted by $S_1$) consists of 50 randomly chosen vertices (where each vertex had equal chance to be included). Randomly chosen vertices represent a realistic situation due to the fact that it is rather hard to successfully infect specific people in a network, assuming there is no basis of trust or any form of good relationship. Therefore a random set is more likely in real life. We can see there are at least a few initially infected vertices in each cluster (Figure 2(a)).

Another set, represented in Figure 3(a) (from now on denoted by $S_2$), consists of 50 vertices that are distributed over only two clusters. These have been chosen to review what the difference is between widely spread and more closely distributed seed vertices. The second set can be interpreted as each cluster being a country, where the vertices within a cluster represent its citizens, and therefore this represents two countries in which infected citizens are, while in the other countries, the citizens are still healthy.

Finally the last set contains 25 vertices that are distributed over only a single cluster. Shown in Figure 4(a) (from now on denoted by $S_3$). This represents a small community of people that are infected and has been chosen to review what the difference in size of initial set of seed vertices is. Simulations with these sets enables us to see different outcomes when reviewing different percentage of infection as well as review the effect of containment in several situations.

5.2 Containment methods

The purpose of this paper was to get more insight in what methods are possible and how effective they are in containing a social network worm propagating through a network as mentioned prior to this section. We define containment as the actions taken with the end goal described as limiting the spread of a virus to the highest degree possible. The effectiveness of a containment method therefore is determined by its success in decreasing the amount of infected vertices at the end of propagation but with the limitations that it brings with it in mind.

5.2.1 Blocking links or vertices

Blocking links in a network is one of the most logical ways to defend your users from getting infected and slowing down the propagation of a worm. If the worm is unable to get to your users there is no way to infect them. The unfortunate part of this kind of containment is that you are in fact disabling part of your network and therefore users from being able to connect to each other. This possibly lowers the user-friendliness of your network.

The choice of which vertices or links should be removed has to be made. There are multiple ways to determine the importance of a vertice or link within a network. These differ per type of network, directed or undirected. [5] determines the importance of a link by applying the bond percolation method. This is a method of finding paths between vertices and reflects the importance of a link between two nodes. However the bond percolation method is not applicable to undirected networks because it is dependent on the direction of an edge and our network is undirected.

To solve this problem a different technique for estimating the importance of a vertice or a edge was used. The betweenness centrality of a vertice or an edge is a score that reflects the centrality of that vertice or edge based on shortest paths. The score is calculated by taking the sum of the number of times this vertice or edge is passed through, when finding the shortest path between two other vertices, divided by the total number of shortest paths between these two nodes.

$$C_B(e) = \sum_{(s,t) \in E} \frac{\sigma_{st}(e)}{\sigma_{st}}$$

Here $C_B(e)$ is the betweenness centrality of edge $e$, $\sigma_{st}(e)$ is the number of shortest paths through edge $e$ for vertice $s$ and $t$ and $\sigma_{st}$ is the number of all shortest paths from $s$ to $t$ [4].

Upon removing edges, starting with the edges that have the highest betweenness centrality score, the graph of the network starts decomposing into clusters. If edges are continuously being removed, at a certain point you start to completely separate clusters from each other which means that if none of the vertices in the individual clusters are infected, the entire cluster stays healthy.

In our research we review how different stages of decomposition relate to decreased infection rates to conclude about the best way to pro-actively prevent users from becoming infected.

The different stages of decomposition are relatively important. Ideally you would want to remove every edge between bridge vertices starting with the most important ones until the clusters separate. The method used however, does not always target the specific edges you want. When removing edges, it is important to realize that you are in fact disabling part of your network from communicating with each other and therefore these measures should not be taken lightly. Removing edges in graphs has been used successfully previous to this research to obtain clusters in a network, but unfortunately there is no point in time where removing edges is optimal. Most clusters consist of smaller clusters, which might consist of even smaller clusters. This means that you can keep removing edges until each vertice in the network has been separated. We reviewed how many edges need to be disabled to save a portion of your users.

To review how far we have to decompose, we looked at a couple of different stages, keeping in mind that disabling edges results in network disturbance. We reviewed in total four different stages and assessed the benefit it might have. The stages are 5000, 10000, 15000 20000 edges removed and are shown in Figure 5. Note that the entire network consists of around 88,000 edges, and therefore temporarily 23% of the network is unable to communicate with each other if we remove 20000 edges. This is not necessarily a bad situation but by far from ideal. Most of the edges that are being removed, consist usually of edges with relatively minimal interaction, for example someone connected to a person they once met on vacation (bridge links), but this is not always the case.

6. RESULTS

6.1 Infection

First, we conducted the infection experiment. For each infection probability $p$ and set of seed vertices $S$ we obtained 100 simulations. All of the following numbers include the initial seed vertices and are also displayed in Table 1, Table 2 or Table 3.

For $p = 1\%$ the results are as following. For $S = S_1$ on average 247.6 people got infected (6.1% of population). The standard deviation was 76.09 with the maximum at 408 and minimum at 69. From Figure 2(b) we can see that the infection propagated within the initial clusters but did
For $S = S_2$ on average 175.47 people got infected (4.3% of population). The simulation had a standard deviation of 52.60 with the maximum at 413 and minimum at 64. This simulation was very similar to the first one (Figure 3(b)). The infection did not propagate outside of the cluster. However, the total amount of infected users was lower compared to the first simulation. This can be explained by basic probability. The vertices are more closely distributed in $S_2$ compared to $S_1$ and there are less directly neighboring vertices that can be infected.

For $S = S_3$ on average 37.34 people got infected (0.9% of population). The standard deviation for this simulation was 21.09 whereas the maximum was 243 and minimum 28. This set of seed vertices had the lowest number of infected users which is the expected outcome because it also had the lowest number of initial seed vertices. The infection did not propagate outside the initial cluster on
average, but has a very high maximum compared to the average. These simulations show how important the structure of the network is for probabilistic propagation models, as well as show fragile the propagation of a worm can be if the probability for infection is very low.

The second infection probability shows a large increase in total number of infected vertices compared to the first. For \( p = 5\% \) the results are as following. For \( S = S_1 \) on average 2200.22 people got infected (55.0% of the population) with a standard deviation of 73.25, maximum of 2369 and minimum of 2015. We can clearly see in Figure 2(c) that the number of people that are infected within the clusters is much higher. Another important factor is that there are some key bridge vertices that have been infected (left and right of the center cluster) which might have had major impact on the propagation of infection.

For \( S = S_2 \) on average 1986.7 people (49.2% of the population) got infected. The standard deviation was 231.93 with the maximum at 2296 and minimum at 957. The biggest change between the prior simulations for this set is that now the infection propagated to neighboring clusters where multiple bridge vertices on the left have been infected (Figure 3(c)) whereas the right side of the graph is still healthy due to its bridge vertices not being infected. This can be explained by the difference in amount of links between the clusters and its bridge vertices. The right two clusters have less connections compared to the left two clusters (Figure 3(a)). For \( S = S_1 \) on average 1247.93 people (30.9% of population) got infected. The simulation had a standard deviation of 896.44 with the maximum at 2254 and minimum of 103. As with \( S_2 \) the infection propagated outside its initial cluster as well as infect many within those clusters but failed to infect the right side of the graph (Figure 4(c)). For this simulation there is a very large difference between the minimum and maximum value of the simulations. This behavior can be explained due to the fact that all of the initial vertices are within one single cluster. In this case the entire propagation relies on the bridge vertices, and because the infection probability is still relatively low there are going to be instances where it does not propagate outside, resulting in a very large difference in maximum and minimum.

Finally the highest infection probability and also the one with the most potential. For \( S_1 \) on average 3068.91 people (75.9% of the population) got infected. The simulation had a standard deviation of 31.32 with a maximum of 3143 and a minimum of 2945. When reviewing the propagation we can see in Figure 2(d) that now every cluster has got many infected vertices, where 5 out of 8 main clusters are nearly fully infected. This clearly shows the danger of these kind of propagations.

For \( S_2 \) the simulation average was 2943.42 with a standard deviation of 117.17. The maximum was 3115 and minimum was 2587. This simulations shows (Figure 3(d)) many regions being highly infected where only two main clusters are fully healthy whereas with 5% infection probability three clusters stayed healthy.

For \( S_3 \) the average was 2728.83 with a standard deviation of 752.07. The maximum was 3115 and the minimum was 191. Like the simulation of \( S_2 \) with 5% the difference between the minimum and maximum is very large again.

### Table 1. Infection results without containment for 1%

<table>
<thead>
<tr>
<th>Set</th>
<th>( S_1 )</th>
<th>( S_2 )</th>
<th>( S_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>24.17</td>
<td>175.47</td>
<td>37.34</td>
</tr>
<tr>
<td>Stdev.</td>
<td>76.09</td>
<td>52.60</td>
<td>21.09</td>
</tr>
<tr>
<td>Max.</td>
<td>408</td>
<td>413</td>
<td>243</td>
</tr>
<tr>
<td>Min.</td>
<td>69</td>
<td>64</td>
<td>28</td>
</tr>
</tbody>
</table>

### Table 2. Infection results without containment for 5%

<table>
<thead>
<tr>
<th>Set</th>
<th>( S_1 )</th>
<th>( S_2 )</th>
<th>( S_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>2200.22</td>
<td>1986.7</td>
<td>1247.93</td>
</tr>
<tr>
<td>Stdev.</td>
<td>73.25</td>
<td>231.93</td>
<td>896.44</td>
</tr>
<tr>
<td>Max.</td>
<td>2369</td>
<td>2296</td>
<td>2254</td>
</tr>
<tr>
<td>Min.</td>
<td>2015</td>
<td>957</td>
<td>103</td>
</tr>
</tbody>
</table>

### Table 3. Infection results without containment for 10%

<table>
<thead>
<tr>
<th>Set</th>
<th>( S_1 )</th>
<th>( S_2 )</th>
<th>( S_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>3068.91</td>
<td>2943</td>
<td>2728.83</td>
</tr>
<tr>
<td>Stdev.</td>
<td>31.32</td>
<td>117.17</td>
<td>752.07</td>
</tr>
<tr>
<td>Max.</td>
<td>3143</td>
<td>3115</td>
<td>3115</td>
</tr>
<tr>
<td>Min.</td>
<td>2945</td>
<td>2587</td>
<td>191</td>
</tr>
</tbody>
</table>

However the maximum is a lot closer to the average compared to the minimum. This shows that even an initial group of infection vertices grouped together in a single cluster, can deal a large amount of damage (Figure 4(d)).

### 6.2 Containment

For the containment method a total of five different stages at which the edge removal commences have been chosen. These stages differ from each other by the percentage of vertices in the network that have been infected and reflect a real life situation where the attack is discovered. These stages range from 10% to 50% of the total network being infected and are therefore either a fast discovery or a slow discovery. Each simulation for containment has run 100 times and the infection algorithm was given the same set of seed vertices that were used with the infection. The infection probability differs from the infection simulations and is either 5% of 10%. The lowest infection probability of 1% was considered for containment as well, but due to its minimal effect on the graph (number of infected people < 10%) was included in the containment experiment.

#### 6.2.1 Moment of discovery

Containment consists of two parts, discovery of attack and performing counter measures. The discovery of an attack can be passive or active. With passive we mean that infected users report suspicious activity and upon investigating the attack gets discovered. This is rather slow because you need at least a certain amount of reports before investigating is worth it. Of course, there are always methods to monitor strange behavior from users, this would be a more active measure.

For the first set \( S_1 \) we can see that when the probability of infection becomes higher, it is less beneficial to have very quick discovery compared to lower infection probability. Figure 6(a) shows this by a very steep slope followed by a rather shallow slope and finally a little increase again. For 5% and 10% infection probability the same behavior is present but for 10% this is shifted to the right. The steep slope happens after 20% followed by the shallow slope as
well as a little increase. For the 5% infection probability the highest increase in infected vertices happens after 10% (Figure 6(a)) while for 10% this is after 20% of the network has been infected.

For $S_2$ the figure looks very logical for the situation. Here the initial vertices are not spread out over all clusters but distributed over 2 clusters (Figure 3(a)). This means that decomposing will be a lot more effective than before with $S_1$. We can clearly see that the effect of removing edges decreases as the number of infected users increases (decreasing slope when moving right on x-axis).

Finally for $S_3$ the results are a little harder to interpret due to the high difference in minimum and maximum value. To obtain a better idea about how the average relates to the minimum and maximum a box plot was made. From Figure 8 we can see that for 5% infection probability there is a decreasing number of vertices that can be saved when the containment method is applied at a later stage. Although

---

**Figure 6. Infection results for $S_1$ after removing edges**

**Figure 7. Infection results for $S_2$ after removing edges**
the averages seem to rise in a relatively linear fashion, the maximum of infected users does not increase as much as it does in the first stages which means the effect of the containment method is not as efficient at later stages. For 10% infection probability a similar behavior is present (Figure 9) but happens with higher number of infected users.

6.2.2 Edge removal

As mentioned the effectiveness of edge removal depends on three things. Infection probability, amount of edges removed and the stage in which the edges are removed. The stage has been mentioned. To understand how infection probability plays a role in this measure we review Figures 6, 7, 8 and 9. For $S_1$ we can see two things (Figure 6). At the beginning removing the next amount increases the number of people that can be saved by a similar amount. This decreases when more people get infected and after 50% of the network is infected the difference between removing 5000 or 20000 edges is very little for 5% infection probability. For 10% infection probability there is a little more benefit.

For $S_2$ there is a very similar behavior between 5% and 10% infection probability. The effect of removing more edges diminishes when a larger portion of the network is infected before applying the method. In the end there is little to no benefit from removing 5000 or 20000 edges while in the beginning there is a large difference. Early discovery can result in a decrease of at least 30% of total number of infected users for a 5% infection probability (Figure 7(a)) when removing 20000 edges or 5000 edges. For 10% infection probability this number decreases to 29% (Figure 7(b)).

For $S_3$ the results of removing 5000, 10000 or 15000 edges is very similar for 5% infection probability. From (8(a), 8(b) and 8(c) we can see that the number of infected users stays about the same at each stage. The only benefit from removing more than 5000 edges can be made by removing 20000 edges before 20% of the network is infected (Figure 8(b)). For 10% infection probability the same is true. Removing 20000 edges before 20% makes the biggest difference where approximately an additional 12% of users can be saved. After this point removing 15000 edges is as beneficial as removing an additional 5000.

7. DISCUSSION

During this research a number of assumptions have been made. In this section we will go through them and explain why these choices were made and on what basis. First of all the dataset used for the simulations is not one of the networks that we specifically are interested in. However, the characteristics that describe the Facebook dataset are generalizable to most social networks. Its edge distribution is power law based and scale-free, it is small-world and highly clustered. Therefore any of the results obtained from these simulations are applicable to any other social network with these characteristics. Of course this is only a sample of the entire network and therefore the results are based on a sample as well.

In Section 6.1 there are multiple figures with infection results. These figures are based on the average of 100 simulations. However we can not make a graph of the average and therefore we chose the simulation with results that were the closest to the average. This results in the figures showing a resemblance, but does not guarantee that these exact vertices would be infected in most cases. Especially for the third set of initial seed vertices $S_3$ the figure should be interpreted lightly due to the high standard deviation.

In Table 4 the number of infected users in the simulation that was used for the graph is given.

For this research the infection probability was the same for each user at all times. How realistic this is, can be
questioned. Usually when a virus is spreading through a network, people would start to tell their friends, maybe it gets in the news, and therefore people start to get aware as more people get infected. This should result in a lower infection probability. Another aspect is that for each cluster the same probability was used, while these exploits are usually language based. If we were to define each cluster as a different country the probability would be different as you generally wouldn’t expect a message in a different language (e.g. a german user receiving an english message).

During the infection simulations, the number of simulations was set to 100. This number was chosen due to the fact that this research had a time restrictions. If the number of simulations would have been higher the research would not have been finished in time. The results are therefore less accurate compared to if a higher number of simulations was used. For the simulations on infection a confidence interval was calculated to review how accurate the results are. For example for 5% infection probability the 95% confidence interval ranges from 0.6% to 14.2% whereas if we had done for example 1000 simulations this could have been lowered to at least 0.2% to 4%. Especially for the third set the results are somewhat hard to interpret due to the high standard deviation.

The 95% confidence interval for the average of these simulations ended up ranging from 1.6% to 2.3% of the real value, whereas this could have been improved to at least 0.05% to 0.7% if we had run the simulations 1000 times. The extra time it would have taken was at least 2.5 times as much depending on the infection percentage and amount of edges that had to be removed.

8. CONCLUSIONS

In this research we analyzed the Independent Cascade model and created a propagation model for a social network worm based on this. We found that there is no generalizable solution for social network worm propagation. The effectiveness of countermeasures depends on the infection probability as well as network structure. Due to specific characteristics (in this case undirected edges) not every containment method is applicable. For the methods that were considered in this research the results showed that in most cases early detection is beneficial to prevent large scale outbreaks but the precise effectiveness is situation dependent on what portion of the network is infected, as well as placement of the infected vertices.

9. ACKNOWLEDGEMENTS

We would like to thank Doina Bucur and Erik Tews for general guidance and providing helpful information as well as clarification and confirmation on several cyber security and graph theory topics.

10. REFERENCES


