Social Network Analysis Project: Models and Analysis of WannaCry propagation

Fangqing Yuan, Francesco Redaelli, Diego Croci

November 27, 2019

INTRODUCTION

The advent of the network of networks par excellence, the Internet, brought with itself a whole new chapter in the fields of social network and epidemics: computer viruses. In the following lines, we propose an analysis of the one of the biggest ransomware cyber-attack of the last years: WannaCry.

A ransomware is a computer virus that encrypts the victim's files, making them inaccessible, and demands a ransom payment to decrypt them; it usually spread through email network and it has the capability of further replicating itself after the infection.

WannaCry propagated through EternalBlue, exploiting a vulnerability in the Server Message Block (SBM) protocol in older Windows systems, namely Windows XP and Windows 7. Although the main outbreak, started on the 12 May 2017, lasted only 48 hours, WannaCry is considered one of the biggest cyber-attack in terms of size, speed of infection and estimated social damages.

Within 2 days, the virus affected more than 250K systems in 150 different countries, hitting both privates and public bodies and paralyzing companies and institutions.

It was eventually found out that WannaCry included code that looked to check if a specified domain had been registered. If it received a response from the domain, it shut down. This mechanism was probably inserted to prevent investigators studying the software in a closed virtual environment called "sandbox" (from within which any URL or IP address will appear reachable)

Marcus Hutchins, a British security researcher, discovered such a mechanism, decrypted the hard-coded URL and paid 10.96 dollar to register the domain and set up a site there, slowing WannaCry spreading dramatically and leading to its death.[1]

MODELS and ASSUMPTIONS

Asking ourselves what would have happened if the so called "kill switch" mechanism was not found, we tried to come up with a model fitting WannaCry spreading, in order to predict how much would the virus spread if unstopped and what could have been done it order to stop it.

The model is based on the results of the paper "Email networks and the spread of computer virus" by M. E. J. Newman, Stephanie Forrest and Justin Balthrop, 10 September 2002 [2]

We introduce further assumptions in order to reproduce the Worst Case Scenario (WCS)

Assumptions:

• Nodes: Computers (Users – Each user has a single PC, used to read emails he/she receives)

• "Address Book" of user i: List of regular correspondents of user i

• There may exist an edge running from user A to user B if B's email address appears in A's address book (directed edges)

• In general, it is not always true that given (1), the reverse holds (A does not necessarily appears in B's address book). However, in WCS that is actually the case (Reciprocity = 1; Reciprocity is a measure of the likelihood of vertices in a directed network to be mutually linked i.e. having double links with opposite directions)

• Graph: Under those assumptions, the graph will be Undirected (WCS)

• At time t, node i is either "Healthy" or "Infected (and Infector)"

• If at time t node A is Infected, node B is Healthy and there's an edge linking A and B, B will become Infected at time t+1 (WCS – B reads the infected email and B has a vulnerable PC)

• There is no such thing as "Healthy Infector" (as it could be, for instance, in human diseases)

• No node can become Healthy again if it has been Infected unless an external planner intervenes (number of people that paid the ransom is negligible – actually this is a reasonable assumption, since such a percentage was about 1/1000 infected Users, and many of them stated they didn't get their files back)

We found out in the literature that e-mail viruses spread following the **Sublinear Preferential Attachment** model. The theory states that, in any preferential attachment model there is some dependence between the **probability of receiving a new connection** from an incoming node and the existing nodes' degree; and actually this is described by the following formula:

$$\pi(k) = k^{\alpha}$$

with $\alpha > 0$. Now, the difference between linear and sublinear lies in the value α, namely: For $\alpha = 1$ then we are in the case of **linear** preferential attachment, while for $\alpha < 1$ it is **sublinear**.

This means that, in the latter case, we always have a relation between some node's degree and the probability of getting new connections, but it is not linear; as a consequence, we will have more "decentralized" graphs where the maximum degree value will be lower (than in linear p.a.) and there are less and smaller hubs.

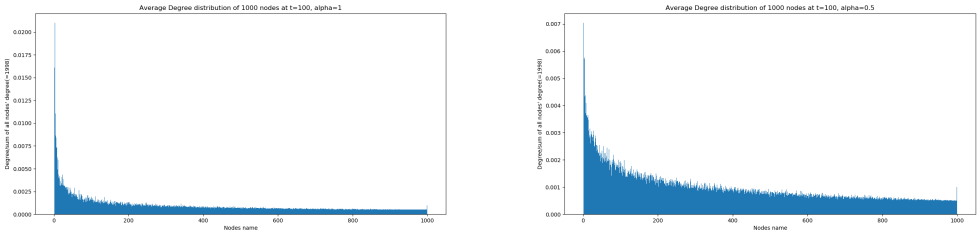

Figure 1: Degree distribution of two preferential attachment model with size=1000. ($\alpha = 1$ on left, $\alpha = 0.5$ on right)

Basic Reproductive Number In order to answer the question "What if nothing was done to stop the virus?", we need to introduce a new bit of theory. We will use as a reference the book "Networks, Crowds, and Markets: Reasoning about a Highly Connected World" by D. Easley and J. Kleinberg.
The Basic Reproductive Number is defined as follows:

$$R_0 = pk$$

. Where p is the **probability of infecting each neighbor** and k is the **individual nodes' degree**.
In the literature we found out that the following holds:
"If $R_0 < 1$, then with probability 1, the disease dies out after a finite number of waves. If $R_0 > 1$, then with probability greater than 0 the disease persists by infecting at least one person in each wave."[3]
Remembering that we estimated the probability of infecting neighbors with the fraction of vulnerable computers (those which used **Windows 7 or XP**) which was around 0.56, we conclude that the virus would have spread if $k > 2$; which is very likely considering the network of computers at that time.

VACCINATION

With $R_0 > 1$, the virus would infect all nodes in the connected graph in a finite time. We come up with a solution to stop this procedure, which is vaccination. Therefore we introduce a new type of node: **Immunized**.

An immunized node is previously a healthy node, and after it becomes immunized, it will not be infected anymore. Thus, it will never be infected or infecting others after vaccination.

Vaccination starting time t:

When there are at least a certain portion of nodes are infected, we start vaccination. The portion $x\%$ is called the vaccination threshold.

Two types of vaccination:

1, Random vaccination: at each time step starting from time t, randomly pick a Healthy, not Immunized node and make it Immunized.

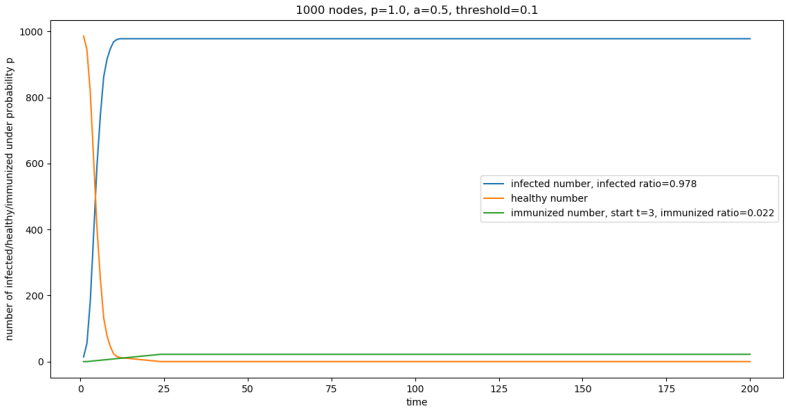

Figure 2: Change numbers of three different types of nodes by time under random vaccination. With $\alpha = 0.5, p = 1, totalsize = 1000, threshold = 10\%$

Not very effective: only 2%of the population gets immunized and the outbreak of the virus is very fast.

2, Targeted vaccination: at each time step starting from time t, pick the Healthy, not Immunized node with the highest degree and make it Immunized.

Very effective: we were able to immunize 10% of the population

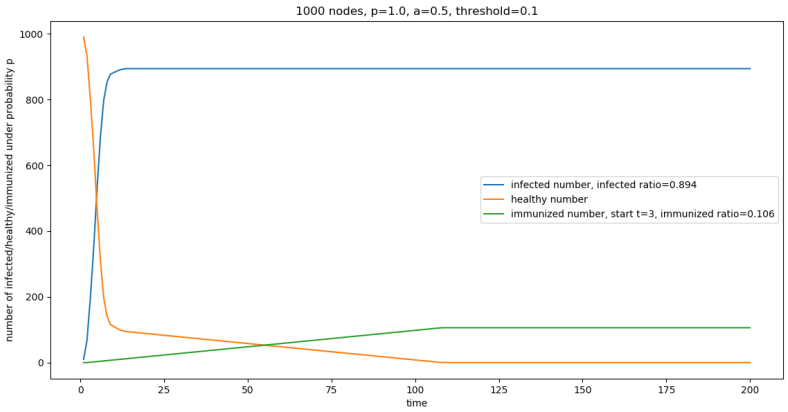

Figure 3: Change numbers of three different types of nodes by time under target vaccination. With $\alpha = 0.5, p = 1, totalsize = 1000, threshold = 10\%$

MODEL REALIZATION BY PYTHON3

We realized all the models spreading, and vaccination procedures by Python. Here are the graphs and code with notes.

```
import matplotlib.pyplot as plt
import random
import numpy as np
import networkx as nx
# let user input 5 parameters
total_nodes_number=int(input('Size of graph, positive integer larger than 1:
    \n'))
prob=float(input('Float, probability of spreading to neighbors, 0<p<=1: \n'))
A=float(input("Alpha for quasi-linear PA model, positive float: \n"))
total_time=int(input('Total time of spreading: \n'))
threshold=float(input('Threshold for immunizing, positive float: \n'))
def run(total_nodes_number,prob,A,total_time,threshold): #main function
   adj_mat=[[0,1],[1,0]] #initial graph adjacency matrix
   i=3
   degree=[1,2] #initial degrees by configuration model
   table={1:1,2:1} #initial degree dictionary
   while i<=total_nodes_number:
      for row in adj_mat:
         row.append(0)
      adj_mat.append([0]*i) #update matrix size from n*n to (n+1)*(n+1)
      weight=[] #weight for deciding which node to link
      alpha=A
      values=list(table.values())
      total=0
      for value in values:
         total+=value**alpha # total=sum of old degrees**A
      exist=list(table.keys())
      for node in exist:
         weight.append(table[node]**alpha/total) #assign weight according to
             sub-linear PA model
      new_link=random.choices(exist,weights=weight,k=1)[0] #choose new link to
          establish
      degree.append(new_link) #update configuration model
      degree.append(i) #update configuration model
      table[i]=1
      table[new_link]+=1 # update degree dictionary
      node_name=new_link
      adj_mat[node_name-1][i-1]=1
      adj_mat[i-1][node_name-1]=1 #update adjacency matrix
      i+=1
   healthy=list(range(2,total_nodes_number+1))
   imm=[]
   infected=[1] #initialize three types of nodes
   time=1
   infsize=[]
   healsize=[]
   immsize=[]
```

```
mark_time=[] #to mark vaccination starting point
while time<=total_time:
   potential=[]
   for nodes in infected:
      i=0
      while i<=total_nodes_number-1:
         if adj_mat[nodes-1][i]==1:
            potential.append(i+1) # find all neighbors of already infected
                nodes
         i+=1
   infection=[]
   for man in potential:
      d=prob
      bi=random.random()
      if bi<=d:
         infection.append(man) #decide whether to infect or not by
             probability p
   for guys in infection:
      if guys in healthy:
         infected.append(guys)
         healthy.remove(guys) # update nodes' types
   if len(table)>0 and len(healthy)>0 and
       len(infected)>=total_nodes_number*threshold:
      mark_time.append(time) # mark vaccination starting point
      mx=0
      for k in table.keys():
         if k in healthy and table[k]>mx:
            mx=table[k]
            sel=k # find the healthy node with highest degree
      print('{},{}'.format(sel,time)) #print vaccination object and time
      healthy.remove(sel)
      imm.append(sel) # update node type
      del table[sel]
   infsize.append(len(infected))
   healsize.append(len(healthy))
   immsize.append(len(imm)) #store numbers of node types
   time+=1
if len(mark_time)>0:
   markt=mark_time[0] # mark vaccination starting time
else: markt=time
plt.plot(range(1,total_time+1),infsize,label='infected number, infected
    ratio={}'.format(len(infected)/total_nodes_number))
plt.plot(range(1,total_time+1),healsize,label='healthy number')
plt.plot(range(1,total_time+1),immsize,label='immunized number, start t={},
    immunized ratio={}'.format(markt,len(imm)/total_nodes_number))
plt.xlabel('time')
plt.ylabel('number of infected/healthy/immunized under probability p')
plt.title('{} nodes, p={}, a={},
    threshold={}'.format(total_nodes_number,prob,A,threshold))
```

```
   plt.legend(loc='best')
   plt.show()
run(total_nodes_number,prob,A,total_time,threshold)
```

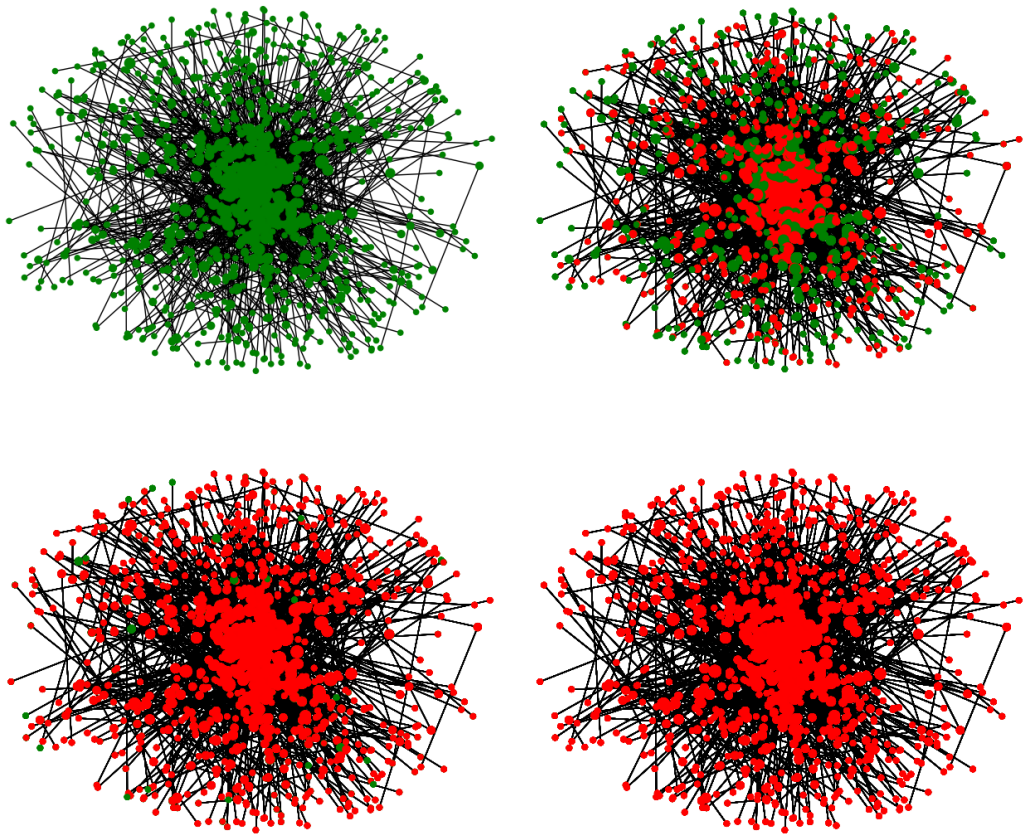

Figure 4: Graph evolution without vaccination. Time= $0, 5, 10, 20$. Green=Healthy, Red=Infected. With $\alpha = 0.5, p = 1, totalsize = 1000$

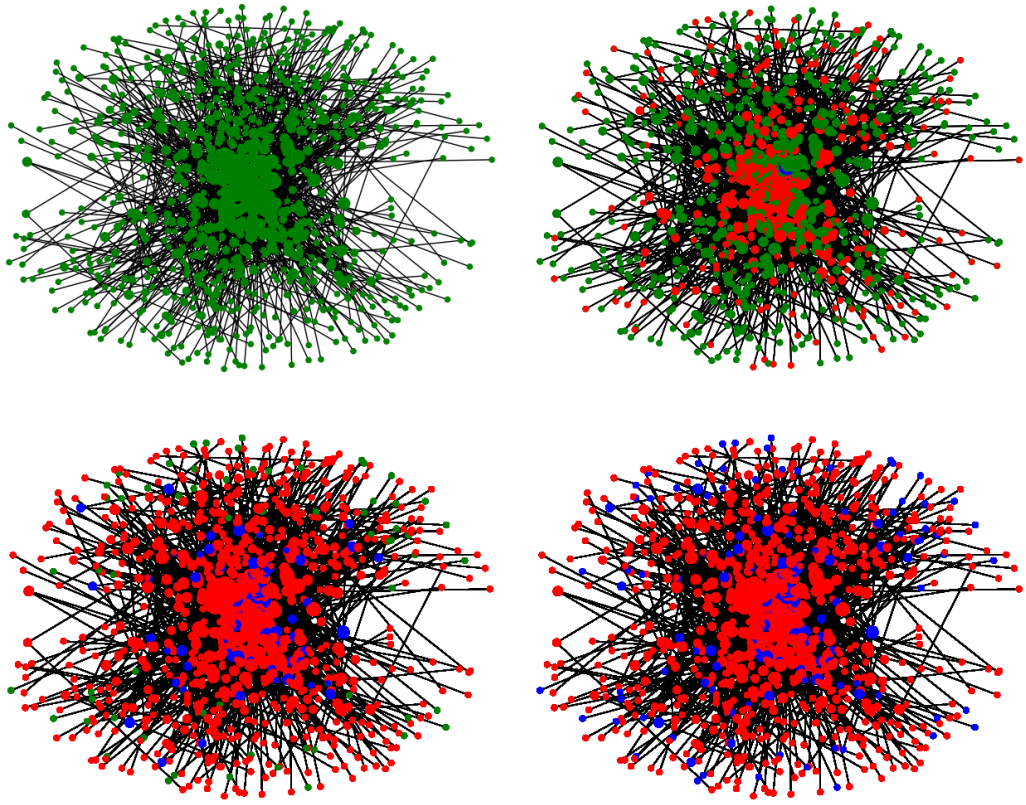

Figure 5: Graph evolution under target vaccination. Time= $0, 5, 76, 200$. Green=Healthy, Red=Infected, Blue=Immunized. With $\alpha = 0.5, p = 1, totalsize = 1000, threshold = 10\%$

CITATIONS

[1]:"What is WannaCry ransomware, how does it infect, and who was responsible?" by Josh Fruhlinger, AUG 30, 2018, https://www.csoonline.com

[2]:"Email networks and the spread of computer virus" by M. E. J. Newman, Stephanie Forrest and Justin Balthrop, 10 September 2002

[3]:"Networks, Crowds, and Markets: Reasoning about a Highly Connected World" by D. Easley and J. Kleinberg. Cambridge University Press, 2010.