Automatically Protecting Network Communities by Malware Epidemiology

Xiao-Si Wang, Jessica Welding and Tek Chung

July 9, 2020

Automatically protecting network communities by systematically analysing the simulation results from malware epidemiological models

Xiao-Si Wang[1], Jessica Welding[1,2], and Tek Kan Chung[1,3]

[1] Applied Research, British Telecommunications plc.
`selina.wang@bt.com`
[2] Lancaster University, UK
`jesswelding@hotmail.com`
[3] Cambridge University, UK
`tekkanchung@outlook.com`

Abstract. Malware epidemiology, especially the modelling and simulation of malware propagation, has been theorised to improve malware outbreak preparedness and drive decision making during real time epidemics. However, practical methods to make use of malware epidemiology are significantly lacking at every level, whether within organisations or at country and global levels. To fill this gap, we present a novel and automatic method to protect networks with a community structure using the malware epidemic final size, one of the most important metrics of a malware outbreak. We treat the final size probabilities abstracted from the simulations as a "signal". We process the "signal" so that the final sizes can be correlated with the communities identified within a network to gain practically usable insights. Finally, we define thresholds and rules built on such insights to deploy automatic protection on the network of concern. To our knowledge, this is the first attempt to make use of malware propagation simulation results as a signal. We show that not only theoretically, but practically malware epidemiology can be used in an automatic manner to protect networks. This study should act as the foundation and inspiration for industrial deployments of malware epidemiology.

Keywords: malware epidemiology · malware propagation · model · simulation · epidemic final size · outbreak severity · signal processing · signal smoothing · rule · threshold · network · community · cluster · stochastic model · agent-based model.

1 Introduction and Motivation

Computer systems are being embedded into the day-to-day activities of modern society at an unprecedented speed. As such, securing computing systems and infrastructure has become more critical than ever. The huge losses and damages from cybercrimes, e.g., WannaCry [3], NotPetya, and Equifax Data Loss, just highlight the importance of robust and agile cyber security. In 2011, Detica/the UK Cabinet Office estimated cybercrime was costing the UK £27Bn annually, about 1.8% of GDP, (£3Bn from individuals, £3Bn from the government and a massive £21Bn from companies [1]. The 2017 Norton report on cyber security presented a shocking figure of £130Bn cybercrime loss globally [6].

All areas of cyber security, from research to field operations, are undergoing rapid transformation driven by technological advances such as automation, big data analytics and artificial intelligence. Many products and tools are being developed to address different areas of security, e.g. end-point, network, cloud, mobile, threat intelligence, and anti-fraud and identity management.

Protecting the networks and computing systems has been predominately achieved through malware or intrusion detection mechanisms which are reactive in nature. This means adversarial events such as malware infection or propagation have already taken place or are taking place if the detection mechanisms are at real time. Therefore it is imperative to introduce predictive measures to the anti-malware arena. Malware epidemiology, especially the modelling and simulation of malware propagation, has long been believed to provide predictive insights and drive decision making, with the first malware epidemiology study published in 1991 [4]. However, agent-based malware propagation models including network-based models are significantly lacking [7] and practical methods to make real world use of malware epidemiology have not been seen.

When producing simulation results from an agent-based network malware propagation model or more generally any stochastic spatial spreading model, because of such models' probabilistic nature in parameter value sampling and the innate spatial heterogeneity, different runs would result in different simulation results. Traditionally and typically whether in malware propagation models or even in infectious disease spreading models, it was to seed the infection source(s)/starting point(s) from different part(s) of the spatial structure and produce multiple simulation runs to see the predicted infected numbers. The multiple simulation results, either randomly selected or selected following some criteria, of predicted infected numbers are subsequently either presented as multiple curves in a plot as they are or just averaged up to show an average curve of predicted infected numbers over time, e.g. covid-19 modelling of non-pharmaceutical interventions [2]. The authors consider two major problems about this approach. First because the complexity introduced by the stochastic model and the spatial structure, the simulation results can be fundamentally different from one another and averaging several runs of the simulation results will hide the heterogeneous infection patterns and will not result in any meaningful conclusion. Secondly this method is very limited to an academic setting in that a plot of several simulations can be presented in an academic paper but cannot simply be used in an industrial setting. In an industrial environment with emphases on automation and on making simulation results usable in a systematic way, simply producing a curve plot does not meet the need. Therefore the authors seek a systematic way of analysing a large number of the simulation runs seeding from all the possible infection starting points, i.e. all the network nodes, so that patterns statistically meaningful can emerge. The authors then carry out a sequence of systematic analytical procedures to analyse these patterns to produce real-world usable results, e.g. statistical analysis of recognised and grouped patterns, on which industrial operational thresholds can be used to set.

Typical predictive metrics of any spreading agent, whether infectious disease or malware, include predicting the epidemic final size, predicting the infection onset time, predicting the infection peak size and peaking time, predicting the epidemic size for a fixed period. We have chosen the epidemic final size, i.e. How big is an outbreak likely to be, because firstly it is an very important predictive metric. In the real world of fighting malware, it means how severe the damage would be if an organisation let a malware runs its natural course without any additional measure. With limited resources and the hundreds and even thousands of attacks to deal with on a daily basis, it is unrealistic to expect all malware attacks being dealt with additional measures on top of the intrusion detection and anti-malware defences in place. Secondly some of the time dependent metrics bring in additional complexity through time, e.g. infection peaking time in some simulation runs will result in one peak but in other runs will result in multiple peaks. Such complexities will distract the focus on this paper which is a first attempt to lead the way to a more systematic procedure to analyse and use simulation results from malware epidemiology.

We will use a simple SIR model as the malware epidemiology model to produce the simulation results. The choice of infection spreading models is not the focus of and less relevant to this study. Whether it is a SIR model or a SIS model or a SEIR model, it does not matter in that all these models will produce the metric we are interested which is the epidemic final size and the choice of spreading model is only for simplicity and for the convention of referring to all infection spreading models as SIR model families and variants.

We will further use a public available social network as the spatial structure to overlay our SIR model on. Contemporary Trojans, such as Emotet, can change its signature down to every half an hour, which makes detection by traditional anti-malware difficult. Making the scenario worse, Emotet propagates by harvesting the social and business networks within an organisation [4]. In an industrial setting, stealthy malware types such as Emotet are one of the most challenging problems to be dealt with on a day-to-day basis. This is to differ from WannaCry and NotPetya which are catastrophic but happen on rare occasions. Therefore using a social network to demonstrate our methods has its own value.

In the following sections, the authors present a novel method to quantify and analyse the simulation results from stochastic network-based malware propagation models. The method processes the simulation results as a signal and summarises the results in a way to directly aid automatic network protections. To our knowledge, this is the first practical and quantitative method to make use of malware propagation simulation results in network protection. As the method developed is complex in nature, we do not use the traditional scientific publish-

[4] https://cyber.wtf/2017/10/12/emotet-beutet-outlook-aus

ing format of methods and results to present it; rather, we discuss the method and an experiment to demonstrate its use in a combined way.

2 Identify the network communities

2.1 The network structure

We first explain several graph terms and how we define nodes, edges and network structures in this paper:

1. Node/vertex
   In graph theory, the terms of node and vertex are used equivalently. We will primarily use the term node, unless stated otherwise.
   (a) A node represents a computing device or system.
   (b) The node changes infection status, e.g. susceptible to infected, infected to removed.
   (c) Each node can also carry attribute information, e.g. node ID, business affiliation, IP subnet.
2. Edge
   (a) An edge connects two vertices, representing that these two vertices have direct interactions between each other which can result in malware infection.
   (b) Each edge can also carry attribute information, e.g. edge ID, which edge connects which nodes, frequency of interactions per edge.

We used a publicly available social network data set to demonstrate our idea, however the application to a social network should not seen as exclusive. Our method applies to any spatial structure in which malware can propagate and communities can be identified. The data set we will be primarily working with is a network of anonymised Facebook friends. This is a network consisting of 4039 nodes with edges connecting those who are friends. The detailed description of this data set can be found in Leskovec et al.[5]. To prepare for further analysis, we need to group all the nodes and edges into different communities by features of the graph. These communities in the real corporate world reflect the lines of businesses, the collaborating teams and the IP subnets the devices are connected into, depending on the information being collected.

2.2 Detect and group nodes into communities

After experimenting several community detection methods, we chose the asynchronous fluid method due to its relatively fast computational speed. The grouped communities of the Facebook data set are shown in Fig. 1, where the different communities are clearly marked in different colours. It is noticeable that some bridging nodes connect the different communities and these bridging nodes will play a vital role in either delaying or stopping the spreading of malware, or helping malware spread into other communities.

To investigate the relationship between different communities and different epidemic final sizes, we need to simulate the epidemic final sizes starting from different nodes within the communities.

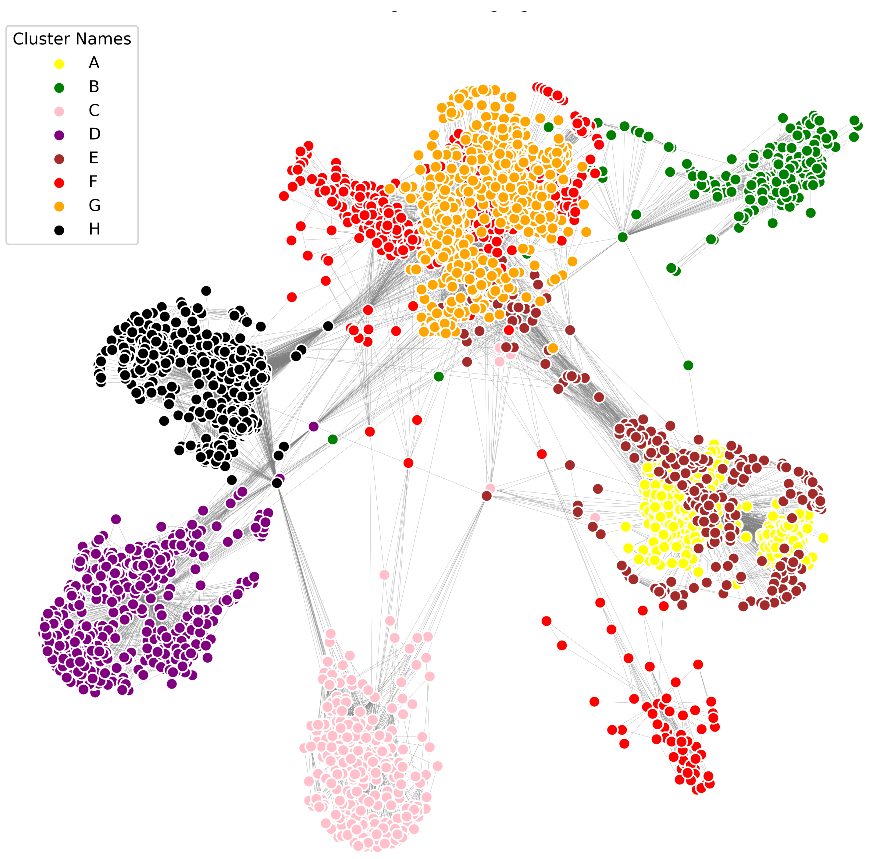

Fig. 1. A visualisation of the communities detected by the asynchronous fluid method in the Facebook data set.

3 Simulate epidemic final sizes using malware propagation models

We performed 20 malware propagation simulations beginning at each node, i.e. patient zero, within this network, with a moderate average node to node infection transmission rate of $\beta_t = 0.05$ and an average removal rate of $\gamma_t = 0.2$ for time unit t. In total this resulted in approximately $80,000$ simulation runs. We used the simple compartmental Susceptible-Infected-Removed model for this experiment, however the methods described can be applied to any spreading model on a network with community structures, e.g. the compartmental status changes can be any SIR variation. Our simulations were run until all the infected nodes had become removed nodes and we recorded the epidemic final sizes for all of the runs. The final sizes are plotted against the epidemic starting node in Fig. 2.

Visibly there is a distinctive pattern that certain starting nodes will result in very different final sizes, e.g. epidemics starting from certain nodes between ID 500 and ID 1000 are quite likely to result in small epidemics around 250 infected nodes. This is hardly seen in epidemics starting from other nodes. It is also worth noting that a significant number of simulations are required to see this pattern, otherwise randomness dominates. With the complexity of the network being studied increasing, the number of simulations should increase accordingly.

4 Smooth the final size probabilities as a signal

4.1 Calculate the probability of each final size

The aim is to group the final size into subsets of high density. For example, if we were to do this visually using Fig. 2, we might decide to split the final sizes into the following ranges: [1] , [2, 100], [100, 250], [250, 3300], [3300, 3600], [3600, 3750], [3750, 4039], each of which exhibits different behaviour.

We first calculate the estimated probability of observing each final size in all simulations. This is a frequency probability calculated by adding up the number of times we observed this final size within the simulations, divided by the total number of simulations. An example of which is shown as Fig. 3.

As stated in earlier sections, the idea is to find the high-density final size ranges that can aid malware infection preparedness. Therefore we are interested in the boundaries of the value ranges. Results shown in Fig. 3 clearly have some kind of noise, particularly around the local minima and maxima. Should we apply algorithms to find the local minima and maxima to the raw data, the ranges found are likely to be affected by this noise. Hence we need to apply smoothing functions to remove such noise.

4.2 Apply a smoothing function

There exist many different algorithms to smooth data, a large portion of which come from the field of signal processing. Although our requirement is different,

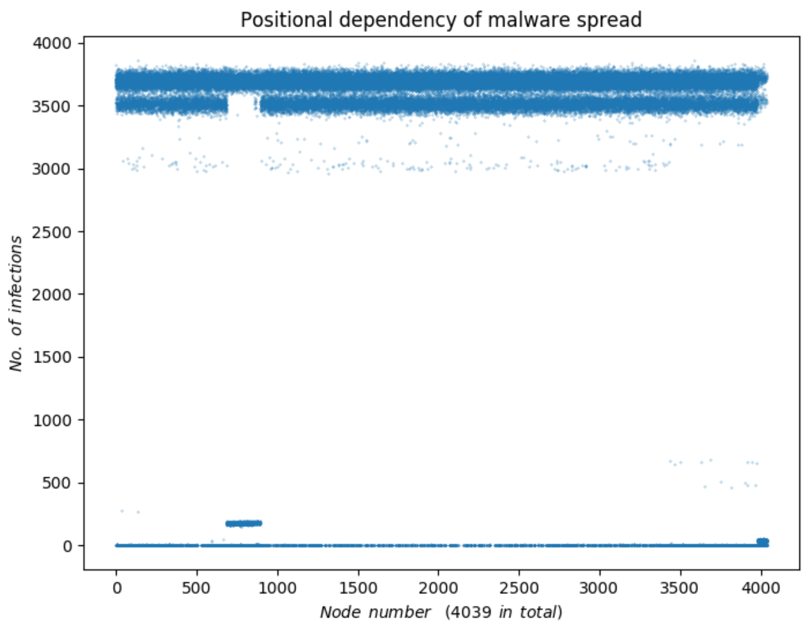

Fig. 2. Positional dependency of malware epidemic final size to the epidemic starting node from approximately 80,000 epidemic simulations. Y axis is the final sizes and X axis is the starting node ID. Note: The node IDs also correlate with the communities they belong, i.e. the node IDs are sequential when the nodes belong to the same community, which makes the patterns more visible when plotted in this way.

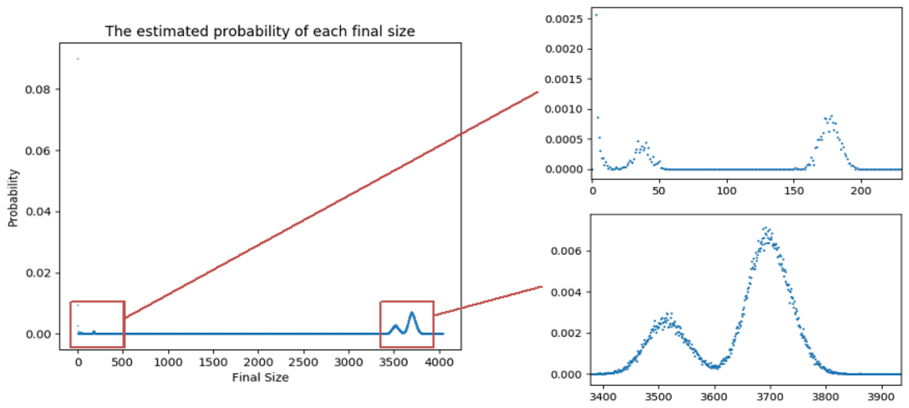

Fig. 3. The probability density dot plot of each final size in the Facebook example. Y axis is the probability of getting a specific epidemic size. X axis is the epidemic size. On the right hand side we include zoomed in high-density probability areas.

we can note that we expect the probability of a final size of "X" to be related to the probability of a final size of "X+1" . As such we can apply these methods to smooth our data.

The aim is to pick out the key features of the final sizes, to be successful in this we need to remove the noise to ensure we do not just pick out random fluctuations. As such we choose to use one-dimensional Gaussian filter to smooth the data. In the experiments, we used $scipy.ndimage.filters.gaussian\_filter1d$ within the scipy module in Python as it produces a good, smooth fit to the data. Additionally, within the methods we have developed we allow for two additional smoothing choices: triangle moving average and a kernel smoother. Together with the Gaussian filter these three functions have been selected as they ensure the curve is smooth and thus we can use methods already developed to find the key features. Throughout we choose to use the Gaussian filter; however, during the process of experimenting different smoothing algorithms, we note that the kernel smoother results in almost identical smoothed data.

The Gaussian smoothing function requires the parameter σ to be selected, with larger σ indicating that we require more smoothing. We expect this to be related to the number of simulations we have run. Additionally we expect that we may always require some smoothing as the true underlying distribution will not necessarily be smooth due to the network structure. Therefore to automate this we choose to set σ as a function of the standard deviation of the data, with a minimum value (σ_{min}) placed so that we always perform some smoothing. Therefore

$$\sigma = max\left(\sigma_{min}, \sqrt{\frac{var(data)}{M}}\right) \tag{1}$$

i.e. the smoothing parameter is the standard deviation of the simulated data divided by the square root of the number of simulations. We take $\sigma_{min} = 5$ for the experiments as this seems appropriate. This choice of σ makes intuitive sense as with greater number of simulations the standard error decreases and therefore we require less smoothing of the data. We plot the smoothed data with the original data in Fig. 4. We can see that the high-density areas and the local minima that separate these areas are still present.

5 Group the final sizes into ranges

5.1 Find all the boundary values marked by local minima in the smoothed probability

To find the subset of high-density and low-density areas, we need to identify the boundaries that separate these ranges. Again, we treat the problem as a signal processing problem but also as an optimisation problem. We use a signal processing method to find all the local minima among the smoothed probability "signal" curve by comparing the neighbouring values surrounding each value. In the experiment, we reverse the curve along the Y axis and use a peak finding method ($signal.find.peaks$ within the scipy module in Python) to find the

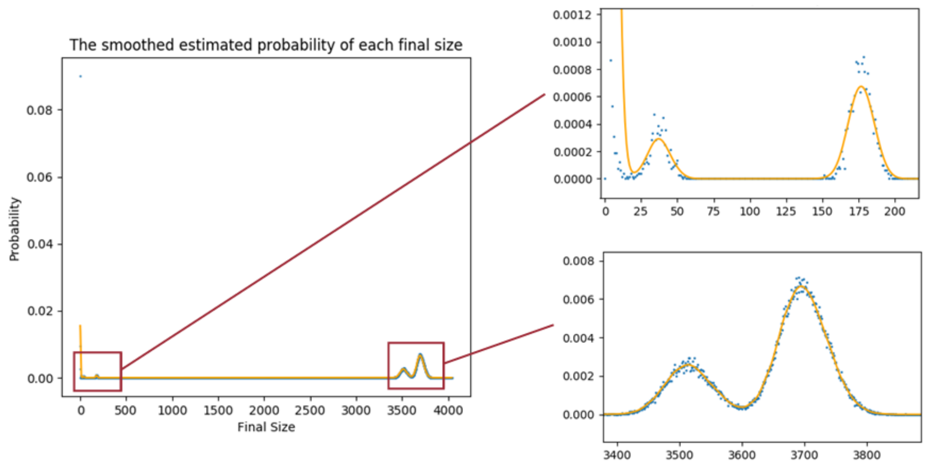

Fig. 4. The smoothing probability curve overlaying the probability density dot of each final size in the Facebook example. Y axis is the probability of getting a specific epidemic size. X axis is the epidemic size. On the right hand side we include zoomed in the high-density probability areas and the smoothing cure of those areas.

maxima and the corresponding x axis values which are the boundaries we are interested in. The boundaries identified are presented in Fig. 5.

5.2 Merge the original ranges using thresholds/rules

Using the above approach alone, the ranges identified can be very short, as seen in Fig. 5. This is as, although we smoothed the curve, there is still noise. Short ranges are not suitable for further analysis and network protection because we are not interested in a final size range with a low probability to reach, therefore we must merge the ranges. In other words, we need to avoid having ranges in which the probability of reaching such a final size range is very small. We achieve this by automatically and sequentially going through the ranges following some thresholds or rules, e.g. maximum 10 groups, range probability values.

In the experiment, we first add up the probability of reaching each value within one range as shown in Fig. 5, e.g. if the range before merging is $[250, 251, 252]$, its range probability is $P(\text{final size } 250) + P(\text{final size } 251) + P(\text{final size } 252)$. Doing this for all ranges, it forms a series of range probability values. We then go through the range probability values; and if the range probability is less than merging parameter M, we then check to see if its neighbour's probability is also less than M, if so then join the two ranges. We choose to set M so that it satisfies $M = min(0.01, M_q)$ and $M = max(0.001, M)$ where M_q is the q^{th} percentile of the range probabilities, usually we use M_{10}. This ensures that if we have small peaks these are still given the chance to be picked out.

Additionally we set a max number of ranges as $r_{max} = 15$. Then if we have too many ranges we increase M by 0.1 and perform the collapsing from scratch

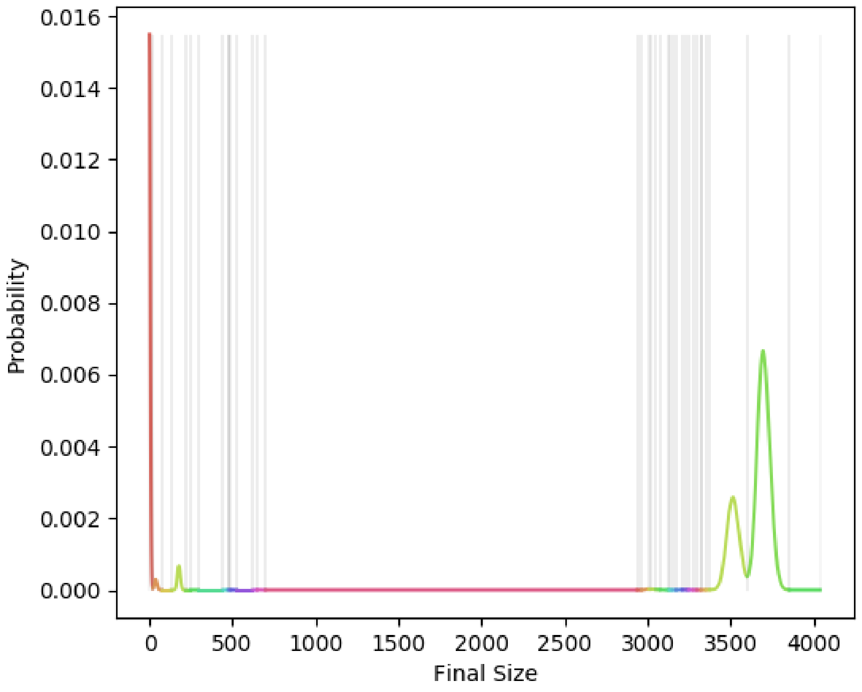

Fig. 5. The high-density and low-density ranges of the smoothed probability "curve" as separated by the boundaries. The boundary values are marked by the vertical lines.

with the new M, this is repeated until we have at most r_{max} intervals. This is to ensure the final output is readable and usable.

After merging the ranges using this method, the final ranges for the experiment are shown in Fig. 6. We can see that there are eight ranges left after merging. These ranges will be analysed together and the community information detected from the network in the next step.

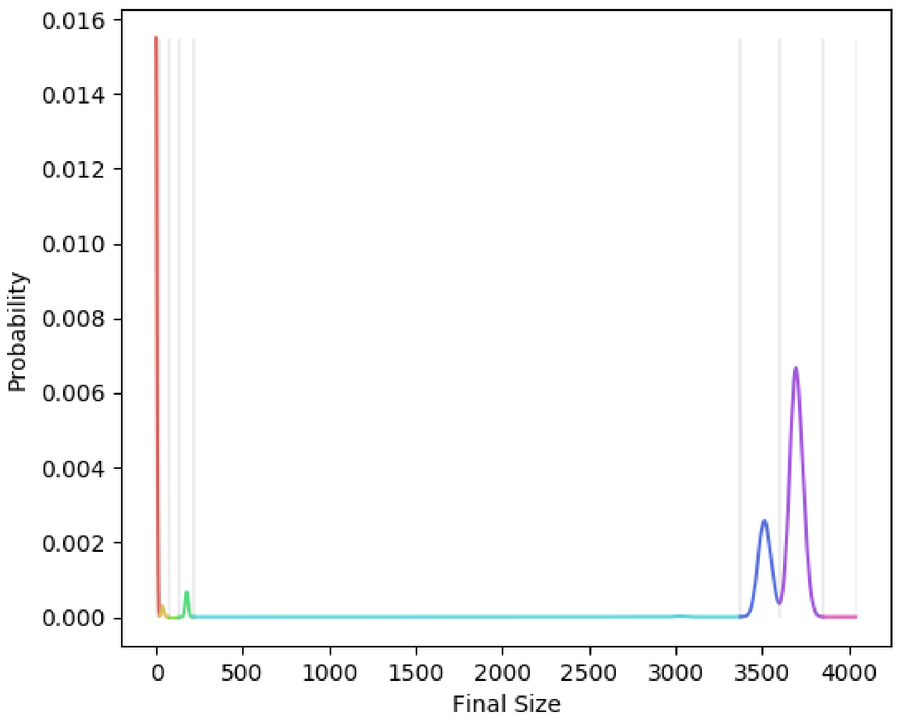

Fig. 6. The high-density and low-density areas of the smoothed probability “curve” as separated by the boundaries. The boundary values are marked by the vertical lines.

How many final ranges for network protection depends on practical needs and what makes sense. We could simply use quantiles of the range probability values to merge. For example, we could find out the 10-quantiles of the range probability values and merge with the cut points defined by 10-quantiles although this simple method may be improved with further rules to insure the ranges are not too small when some of the cut points defined by quantiles are very small values.

6 Identify granular community information/characteristics through relationship analysis between final size range and community contributions

We next aim to find a concise way but meaningful way to make real-world use of the findings. Until now, both the communities introduced in Section 2 and the final sizes (Sections 3 to 5) have been grouped automatically. We can therefore now carry out a number of different types of analysis which quantify the relationships between communities and the epidemic final sizes, e.g. the percentage contribution of each community of nodes to the different final size ranges, the communities which resulted in the most infection cases. We use the experiment to demonstrate how to analyse such data and draw useful knowledge which can inform on thresholds and rules to be used for network protections.

6.1 Epidemic starting community and final size

The number of nodes within each community from the Facebook data set is presented in Table 6.1. There are eight communities detected within the network, each containing hundreds of nodes.

Table 1. The number of nodes within each community.

Community Name	A	B	C	D	E	F	G	H
Number of nodes	553	209	348	545	267	525	840	752

For each distinct final size range and for each epidemic simulation starting community, we calculate the frequency relative to approximate 80,000 malware propagation simulation runs, i.e. the relative frequencies, which are shown as percentages in Table 6.1. Such analytics provide a way to quantify the damage of malware infection in terms of the total number of infected systems, should there be no additional measures to stop the malware from infecting the machines but only the routine removal rate. Certain quantitative community epidemic knowledge can be drawn from the contribution probabilities to the epidemic final sizes presented in the table below:

– All of the communities have at least a 50% chance of producing an outbreak of greater than 3600 individuals.
– Communities A, G and H have the largest chance of producing a severe (> 3600) outbreak. In other words, these communities are prone to large scale spread within the community and therefore are considered more vulnerable to malware attacks.
– Communities B and F contribute to different rare final size ranges which means they are isolated communities which are more resilient to malware

attacks. Malware attacks are less likely to spread out from them if started there or are less likely to spread into them if not started there.

To summarise, Table 6.1 informs us about the relationship between the final size and community in which the outbreak begins.

Table 2. The probability of observing a particular final size, given the outbreak begins in each community represented in percentage.

	A	B	C	D	E	F	G	H
Final size range				%				
[0, 19]	3.4	17.4	23.2	14.6	16.6	14.3	6.1	4.8
[20, 75]	0.0	0.0	0.0	0.0	0.0	4.5	0.0	0.0
[76, 129]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
[130, 217]	0.0	29.0	0.0	0.0	0.0	0.0	0.0	0.0
[218, 3374]	0.5	0.0	0.3	0.2	0.2	0.3	0.3	0.3
[3375, 3599]	27.4	0.4	20.5	24.1	24.0	22.2	27.2	27.2
[3600, 3852]	68.8	53.2	56.0	61.0	59.1	58.7	66.4	67.8
[3853, 4039]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

7 Protect network communities automatically

7.1 Set up rules and thresholds

The ultimate goal is to protect the network communities automatically using the insights gained from the algorithms presented in the preceding sections. As described above, modelling and simulation results have to be summarised into probabilistic findings which can be used to inform the appropriate protection of the network communities. In particular, any values can be used to set up rules or thresholds so that once rule conditions are met, protective measures including prevention and intervention measures are deployed automatically, e.g. re-routing, alerting choking nodes, increasing traffic scans on choking nodes. Such protective measures can include but are not limited to: an anti-malware facility; a malware filter; a malware detector; a block, preclusion or cessation of interaction; and a reconfiguration of one or more computer systems.

Here is one example on how to use the experiment results above to set up rules:

- If
  (a final size >= a threshold value, e.g. >= 3600 infected cases; >= 60% of total systems
  AND
  a probability of reaching such a final size given a starting community >= a certain percentage, e.g. probability of reaching >= 3600 infected cases >= 60%)

– Prepare for malware attacks and deploy protective measures to those communities should a malware be detected, e.g. raising alerts/increasing traffic scans to the users or user communities to be most likely affected.

It is worth noting that we presented a fixed value for the threshold to be set in the above example and a more systematic approach would be to look at the cost function and balance out the cost of a large outbreak and the cost of potential disruptions by any network protection measure, e.g. air-gapping will protect systems but will also stop systems and network to function normally and incur cost.

8 Discussion

In this paper, we describe a quantitative and automatic method of making use of simulation results from stochastic network-based malware propagation models. We show that although the simulation results are complex and difficult to quantify, significant progress can be made by analysing the simulation results as a signal. There are many steps in the method presented in this paper; each step also includes a number of sub-algorithms. We summarise the major steps as follows:

1. Identify the network communities
2. Simulate epidemic final sizes using malware propagation models
3. Smooth the final size probabilities as a signal
4. Group the final sizes into ranges
5. Identify community epidemic information/characteristics
6. Protect network communities automatically with rules and thresholds

There have been many theoretical and academic studies to address malware epidemiology, however the majority focus only on demonstrating the possibility of modelling. For studies which focused on stochastic models, the results presented were usually simulation examples which cannot be directly used in real world protections [8]. Hardly any research looks into analysing all the simulations in a collective way. In an ideal and simple world, the problems can be modelled using mathematical methods with analytical solutions. However, usually this is not possible as malware propagation on a large network consisting of many nodes and edges cannot be modelled analytically. We seek stochastic models and simulation results to gain insights in the complex world.

We use epidemic final size as an example to demonstrate how to analyse modelling and simulation results quantitatively and subsequently use such results to serve protection purposes. The general method of treating the simulation results as a probabilistic signal are generalisable to using simulation results on other key measures of malware epidemics or even other spreading agents, e.g. infectious disease spreading. We acknowledge that the network information is not always easy to obtain, but with data becoming increasingly accessible and advancements in data sharing technologies [9], we expect to see more information on large and complex networks become available for modelling and simulation. We also acknowledge quantifying simulation results focusing on different measures of epidemics will require different considerations, but we argue that treating simulation results as a signal is a good way to identify patterns and to develop methods for the automatic deployment of network protections. We also acknowledge that this method is usable only on any network or spatial structure in which communities or clusters can be identified, which means largely small-world types of networks which are probably mostly social or business networks in the malware propagation scenario and the population mobility networks in the human infection spreading setting. Nevertheless Trojan malware that uses social and business networks to spread represents 70%-80% ongoing malware in the contemporary malware world [5] and therefore we argue that the method is largely applicable to the real world.

To our knowledge, this is the first attempt to make use of malware propagation simulation results as a signal to aid network protection. We show that not only theoretically, but practically malware epidemiology can be used in an automatic manner to protect networks. This study should act as the foundation and inspiration for industrial deployments of malware epidemiology.

9 Declaration

The views expressed in this paper are solely those of the authors and do not necessarily represent the views of their employers.

References

1. Detica: The cost of cyber crime (2011), `https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/60943/the-cost-of-cyber-crime-full-report.pdf`, last accessed 21 April 2020
2. Ferguson, N., Laydon, D., Nedjati Gilani, G., Imai, N., Ainslie, K., Baguelin, M., Bhatia, S., Boonyasiri, A., Cucunuba Perez, Z., Cuomo-Dannenburg, G., et al.: Report 9: Impact of non-pharmaceutical interventions (npis) to reduce covid19 mortality and healthcare demand (2020)
3. Ghafur, S., Kristensen, S., Honeyford, K., Martin, G., Darzi, A., Aylin, P.: A retrospective impact analysis of the wannacry cyberattack on the nhs. NPJ digital medicine **2**(1), 1–7 (2019)
4. Kephart, J.O., White, S.R.: Directed-graph epidemiological models of computer viruses. Proceedings. 1991 IEEE Computer Society Symposium on Research in Security and Privacy pp. 343–359 (1991)
5. Leskovec, J., Mcauley, J.J.: Learning to discover social circles in ego networks. In: Advances in neural information processing systems. pp. 539–547 (2012)
6. Norton: Norton cyber security insights report (2017), `https://www.nortonlifelock.com/content/dam/nortonlifelock/docs/about/2017-ncsir-global-comparison-united-kingdom-en.pdf`, last accessed 21 April 2020

[5] https://www.cisecurity.org/blog/top-10-malware-january-2019

7. del Rey, A.M.: Mathematical modeling of the propagation of malware: a review. Security and Communication Networks **8**(15), 2561–2579 (2015)
8. Wang, P., González, M.C., Hidalgo, C.A., Barabási, A.L.: Understanding the spreading patterns of mobile phone viruses. Science **324**(5930), 1071–1076 (2009)
9. Wang, X.S., Herwono, I., Di Cerbo, F., Kearney, P., Shackleton, M.: Enabling cyber security data sharing for large-scale enterprises using managed security services. In: 2018 IEEE Conference on Communications and Network Security (CNS). pp. 1–7. IEEE (2018)