J Comput Virol (2007) 3:93–101
DOI 10.1007/s11416-007-0051-3

EICAR 2007 BEST ACADEMIC PAPERS

How good is good enough? Metrics for worm/anti-worm evaluation

Attila Ondi · Richard Ford

Received: 12 January 2007 / Accepted: 28 March 2007 / Published online: 21 April 2007
© Springer-Verlag France 2007

Abstract Self-replicating code is a huge problem worldwide, with worms like SQL/Slammer becoming pandemic within minutes of their initial release. Because of this, there has been significant interest in worm spread and how this spread is affected by various countermeasures. However, to date, comparative analysis of spread has been carried out "by eye"—there exist no meaningful metrics by which one can *quantitatively* compare the effectiveness of different protection paradigms. In this paper, we discuss several possible metrics for measuring worm spread and countermeasure effectiveness. We note that the "correct" metric for comparative purposes will vary depending on the goal of the defender, and provide several different measures which can be used to compare countermeasures. Finally, we discuss the idea of significance—that is, what changes induced by worm design or countermeasures are actually *meaningful* in the real world?

1 Introduction

In order to make meaningful comparisons between two different things, people need to understand exactly how these things are being compared. Put more formally, in order to measure something in a way which is meaningful to others, one needs agreed-upon metrics. This is true in all the sciences: measurements and measurement techniques are foundational to the scientific process. However, when one views computer security as a science, the topic of metrics is conspicuously lacking from many discussions. This is particularly true in the field of anti-worm techniques, where there are no agreed-upon methods for comparing the efficacy of different protection mechanisms.

In this paper, we will examine possible metrics used to evaluate anti-worm software. The genesis of this work was a discussion regarding distributed worm defense systems. In comparing two techniques, for example, how can one determine which is "better"? What does "better" actually mean? Should the system be measured on how many machines get infected? On how much damage there is to the network? On the peak doubling time of the infected population? This paper seeks to outline the most important drawbacks with each of these techniques and bring the underlying issues into relief. We conclude by proposing a metric for comparing different protection methodologies from a global perspective.

2 Motivation and related work

To be able to meaningfully measure something, we need to identify what is the purpose of the measurement. On the one hand, it is obvious that whatever the objectives of the worm author, the objective of any reasonable countermeasure is to stop, or at least delay, the worm from causing "damage". On the other hand, it is not at all obvious exactly what one means by the loose and subjective term "damage". Since the individuals, who write the worms, have many different goals they seek to achieve with their creations [2,3] (ranging from curiosity to cause as much havoc as possible), the only meaningful way to compare the efficacy of defense is from the standpoint of the community under threat.

The most intuitive "metric" for measuring the spread of worms is the total number of infected computers over time. Indeed, the majority of the articles written about worms use this technique as the base of comparison (see for example [7,9,13]), even if the authors do not quite realize it! Note

A. Ondi · R. Ford (✉)
Florida Institute of Technology, 150 W. University Blvd.,
Melbourne, FL 32901, USA
e-mail: rford@fit.edu

A. Ondi
e-mail: aondi@fit.edu

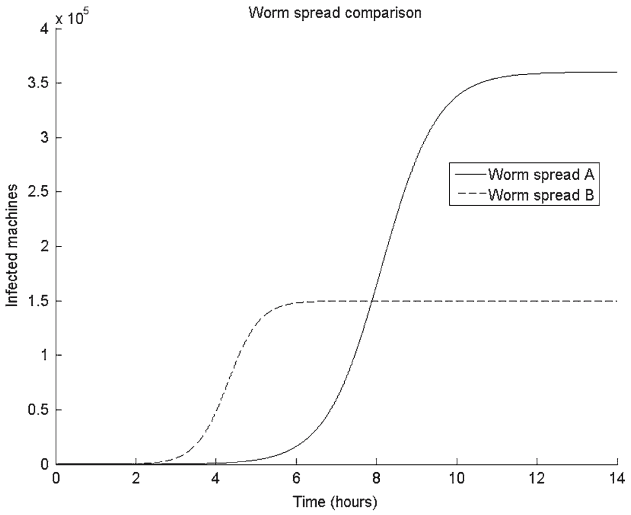

Fig. 1 Spread-curves of two hypothetical worms

that the result is not one number, but a plot of the spread. Comparing spread-plots, aside from the most trivial cases, is not at all obvious. The approach is initially attractive, but limited. Consider, for example, two models for malware prevention (see Fig. 1). The first (we shall arbitrarily label it A) results in widespread infection, but the initial growth of the outbreak is very slow. The second (labeled B) has a very fast growth rate during the outbreak, but the total number of hosts affected is smaller than in A. Which prevention technique is better? In this example, the efficacy of A will depend on *other* processes operating with the system. If the slowdown is large, compared to the reaction time of other detection techniques, A may be preferred; if it is not, then the "better" solution is B. This simple observation is the crux of the paper: the real-world effectiveness of an approach is not governed in vacuo. Both the speed of spread and the size of outbreak are important and metrics need to address both of these quantities.

2.1 Saturation time as a metric

Many approaches to measuring effectiveness are based on "saturation time". We use the term intuitively in this section, loosely meaning the time it takes the worm to infect computers up to the point where there are no more infections. We will discuss a more formal definition below.

For the community, worm A is worse than worm B, if it takes less time for the former to reach its saturation (without any intervention), because less time to saturation means the community has less time to react and reduce the worm's impact. We have to point out that—as many real-life examples showed, (e.g. SQL/Slammer)—reaction time has a minimum value, under which it is impossible to stop the spread of a worm without automatic intervention. Therefore, all worms that can infect all of their susceptible machines under this threshold (*ceteris paribus*) are equally bad, without further differentiation on their exact saturation time. Also, if the total number of computers susceptible to worm A is higher than that of worm B, worm A is more undesirable for the community, because that means more potential infections (especially if we put countermeasures aside), and more work (overall) on cleaning the infection. Moreover, if worm A generates more traffic during its spread than worm B, worm A is worse than worm B for the community.

The logic here is that if the worm generates a high volume of traffic—even possibly saturating the whole network (e.g. SQL/Slammer)—then the throughput available to the community is reduced, introducing massive packet loss and delays in productivity. Some worms also have a destructive payload and while such payloads are not common, they can be devastating to the community. Unfortunately, it is hard to decide in general which payload is more harmful than others. Is erasing your disk worse than encrypting it with a 1,028 bit key? Is turning your machine to a bot better than rebooting it every once in a while? Measuring the dangerousness of the payload depends highly on your individual priorities and assets. Thus, for the purposes of this paper, we assume that all worms have the worst possible payload, particularly as few defense mechanisms are payload dependent.

Measuring the effectiveness of anti-worm techniques is even harder, since one countermeasure may work better against one type of spread, while achieving poor results against other kinds of worms. Calculating a (possibly weighted) average will not help much, as one defense mechanism is unlikely to protect us against all types of worms. If we want to apply layered defense, we need to know how each defense technique performs against various spread types. This implies an agreement on a standard set of spreading algorithms that are applied against the proposed countermeasures. Unfortunately, this has several drawbacks.

First, if we evaluate the countermeasures against a static set of spread techniques, we will not be able to tell how the proposed defense mechanisms perform against new spread types, or even against slight variations of the ones included in the test set. This implies that the test set should incorporate as many spread techniques as possible, including variations. Unfortunately, making such a collection of spread algorithms public increases the risk of new worms being created that use these methods. The new worms only pose a threat until the community employs strong defense techniques against the known spread techniques. Given that the number of proposed solutions defending against vulnerabilities not addressed by existing off-the-shelf protections is high, our time of risk can be expected to be quite high.

Second, using a known set of spread techniques also increases the chance that researchers tailor their proposed counter-techniques specifically towards the test-set. On the one hand, this is counterproductive, especially if the set does not include variations, since custom-tailored algorithms

generally perform poorly against problems they were not designed for. On the other hand, if the test-set includes every known variation, such custom-tailoredness is hard to achieve economically, which encourages general anti-worm techniques. An overwhelming number of articles related to virus- and worm-spread use the "comparison by eye" method [4,5,8,9,11,13]. In this method, the researcher looks at the resulting number of infections over time plots of experiments and, if the curves are intuitively "different enough", he or she draws the conclusion whether the results are in favor of the proposed method. There is hardly any discussion on what difference would be significant enough to draw the conclusion whether the hypothesis is true or not. In fact, these studies generally do not even state their statistical hypotheses in the entire paper, and most of the time they refer to the implied hypothesis that their method is "better" than some baseline. Using statistical hypotheses would require researchers to think about the experiment beforehand and to determine what statistical method is appropriate to test the hypotheses, based on the metrics used to measure the outcome of the experiments. Statistical methods also require researchers to run the experiments several times (in the case of Monte-Carlo simulations, for example) to calculate the variance of the results. Even authors of excellent papers, such as [7,10,12], failed to report the variance of their data, or mention that they performed their simulations more than one time for the experiments.

Current literature implicitly uses the *total number of infected machines at saturation* metric [8,9,12]—for further definitions and discussion see [1], which is in agreement with classical epidemiology studies. From a biological perspective, this might make sense, but from a computer security perspective it does not—other factors, such as peak network traffic, cascade failure, or Distributed Denial of Service (DDoS) come into play. Once a certain number of machines are in the control of the "bad guys", the war is essentially lost.

Surprisingly, no discussion has been formulated about the validity of this metric, or what other metrics we could use in its place. Thus, in this section we identify and briefly describe several possible metrics, then discuss their advantages and disadvantages in detail.

Some of the following metrics are defined in terms of saturation of the spread. The intuitive description of this is that a worm spread reached saturation when the spread curve is (near) flat. We discuss a formal definition in the next section.

In analyzing spread, there are some obvious ways of displaying infection information. Here, we list the most common ones.

2.2 Number of infected machines over time

This representation shows the number of machines infected at every inspected time interval. Choosing the right measuring intervals is up to the researcher. The number of machines infected at any given point in time can be interpolated from the known measurements. Plotting the number of infected machines on the y axis and time on the x axis, the traditional shape of the curve can be approximated to a logistic curve (see Fig. 2a).

2.3 Number of new infections over time

This representation pictures the number of computers that are known to have become infected after the last measurement point—essentially, it is the rate of change of the total number of machines infected. Choosing the measuring intervals is left to the researcher. Plotting the number of new infections on the y axis and time on the x axis, the traditional shape of the curve resembles the bell curve (see Fig. 2b).

2.4 Infectious traffic over time

This representation concentrates on the traffic generated by the spread of the worm, as opposed to the number of machines infected, or the time the spread takes. For most worms, the generated traffic is proportional to the total number of active infections. However, for some distributed (see Fig. 2c) blocking/partitioning solutions, this may not be the case.

Sample graphs are shown in Fig. 2. Using these graphs, the following possible metrics can be derived.

2.5 Number of infected machines at saturation

This metric gives the number of computers infected as seen when the spread of the worm has saturated. This metric is equivalent to the value of the *number of infections over time* plot at the time of saturation. Under this metric, the more machines infected, at the end of the spread, the more successful the worm.

2.6 Total number of infected machines by saturation

This metric is similar to the *number of infected machines at saturation* metric, but incorporates any computer that became infected and have been cured (or became otherwise non-infectious) afterwards, before saturation. According to this metric, the more machines infected, the more dangerous the worm.

2.7 Maximum rate of infections

This metric reports the highest number of new infections between any two measurement points. The metric is equivalent to the maximum value of the *number of new infections over time* plot. Under this metric, the higher the maximum, the more successful the worm.

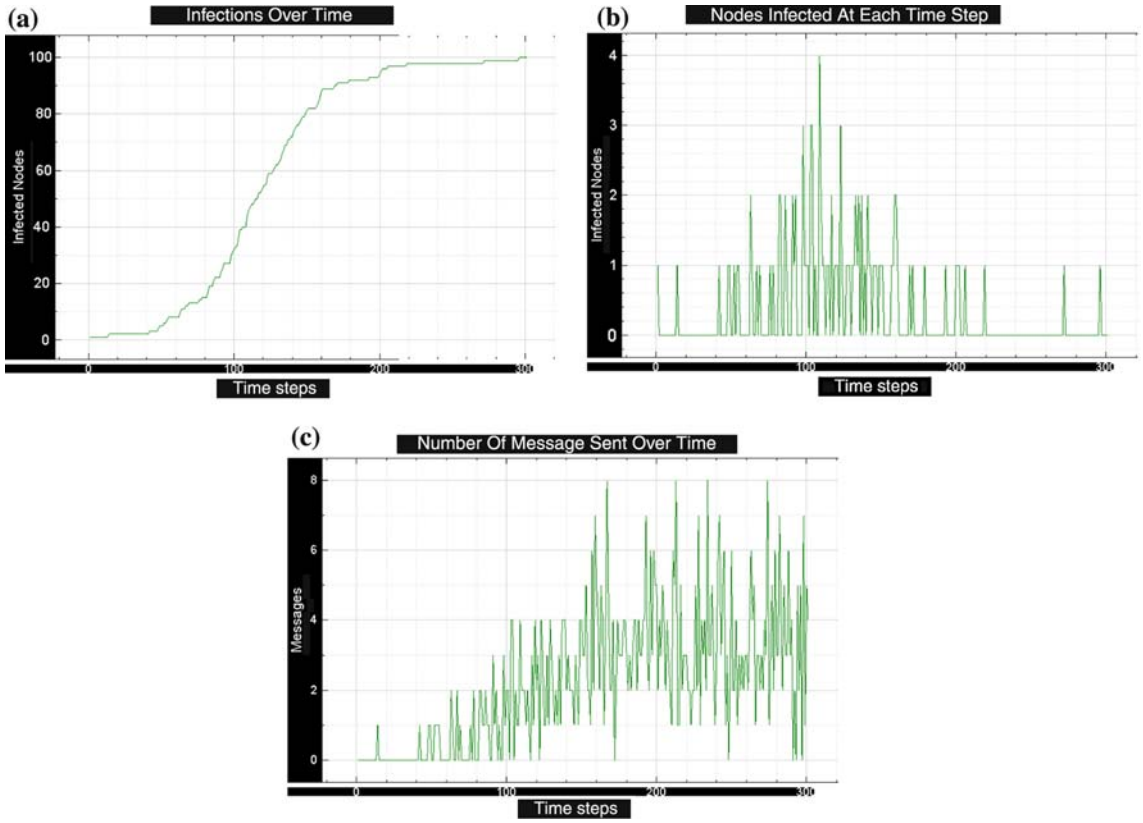

Fig. 2 Different representations of the same spread: **a** number of infected machines over time **b** number of new infections over time **c** infectious traffic over time

2.8 Area under curve

This metric reports the area under the *number of infections over time* plot. Note that, mathematically, the area under the curve is only meaningful when applied to continuous functions, which is unrealistic for real-life or simulation measurements. The exact value of this metric can be approximated in these cases by well known integral approximation methods. Also, to have this metric report meaningful values, the time of the measurements must be bounded. The lower bound can be the time of the initial infection, whereas the upper bound could be the time of saturation (a questionable choice, though see the paragraph *infra*) or the time for the infected population to drop to zero due to machines being cured. Intuitively, the *area under curve* is related to the average length of time a computer is infected and the number of computers infected during the spread. Under this metric, the higher the area under the curve, the more successful the worm.

Note that when comparing two spreads by the *area under curve* metric, the lengths of time of the two measurements and their resolutions must be equal, otherwise a slow spreading malcode would have a large value, whereas a fast-spreading one would have a lower value, even if it infected more machines at the end. By extending the length of the time of measurement of the faster worm, the machines infected earlier would count with a larger weight. This would require recording the length of the measurement and the number of computers infected at the end of the measurement along with the actual area value for comparability with later spreads.

2.9 Fraction of susceptible machines infected

The metrics in this family of measurements are similar to the members of the *number of infected machines…* metric family. However, instead of the sheer number of infections, these metrics report the fraction (between 0 and 1) of the susceptible machines that became infected. The metrics aim to balance the fact that the number of susceptible computers differs highly, depending on the vulnerability exploited by the worm.

2.10 Time to infect all susceptible machines

Instead of concentrating on the number or portion of machines infected, this metric reports the time it takes for the worm to infect every susceptible machine. Using this metric, a worm is considered more dangerous if it takes less time to infect every machine.

Note that this metric requires the knowledge of the number of all susceptible machines. While this is not a problem in simulations, determining the exact amount of susceptible computers in a real-life spread is troublesome. The difficulty is caused by the countermeasures that are put in place against the worm, causing some of the initially vulnerable computers to become protected to the threat.

2.11 Minimum doubling time

This metric reports the minimum amount of time required for the worm to double the population of infected machines. The metric tries to capture the intuition that the faster a worm spreads, the more efficient it is.

2.12 Time to saturation

This measurement is similar to the *time to infect all susceptible machines* metric, except that this metric measures the time it takes the worm to reach saturation, instead of waiting for the whole susceptible population of computers to become infected (which may never happen). The lower the saturation time, the more effective the worm.

2.13 Total infectious traffic

This metric gives the sum of all the traffic during the spread. Since benign users and the malcode use the same network as communication medium, the higher the traffic, the less the others can utilize, therefore, the more efficient the worm. It is true that low network activity can help the worm to go undetected, therefore spread to more computers, but we would require a different metric to capture this aspect of the damage.

There is a problem with this metric similar to the *area under curve* metric, namely, there needs to be a bound on the time this metric is measured. The resolution to this problem is identical to the resolution for the *area under curve* metric.

Looking at the above listed metrics, there is one common property in the use of all the metrics: the more "damage" the malcode causes the community, the more effective it is considered. Another underlying assumption in the above metrics is that every computer (or network link, in case of traffic measurement) has equal weight. In other words, each and every computer is equally important. One might argue with the validity of this assumption, however, there is no easy way of distinguishing more important computers from less important ones. For example, what are the criteria to determine which computers are more important? If we are running simulations, how do we identify the location of the more important machines? Whether they are vulnerable to the worm's exploit is yet another question that needs to be answered.

3 Saturation time: a formal definition

In the previous sections, we used the term "saturation time" loosely. If we want to use it in any metric, we need a more formal definition. Providing such a definition is no easy task. Our definition has to be unambiguously decidable, and in agreement with our intuitive understanding. Unfortunately, our intuitive meaning is heavily based on the common, but most simple, S-shaped spread-curves. Other (perfectly valid) spread-curves can be crafted that break every part of the intuitive meaning, one-by-one. Figure 3 shows some possible spread curves that are unusual in some respect.

If we define saturation time as the time when the worm infects every computer it can, taking into account any anti-worm technique acting against the worm (therefore no more new infections are possible), then determining when this condition arises is a hard task. We need not only understand the spread-technique of the worm, but every possible interaction between the spread of the worm and the applied countermeasures as well.

So far, considering real spread data of worms like Code-Red and SQL/Slammer, we could unambiguously tell from hindsight when the worm presented no more serious threat to the users of the Internet, due to the countermeasures and patches applied. Unfortunately, this cannot always be applied to computer simulations of arbitrary worm spread, because of the possibly random nature of the simulation (as in the case of Monte-Carlo simulation), and the possibly complex interaction between the spread of the worm and the countermeasures applied against it. Even in the case of real-life worms it can be impossible to determine a single point in time, when no more computers could be infected by the worm, due to new computers joining the Internet, reinstallation of computers (which might lack the necessary patches for some time, until these patched are downloaded from the appropriate update site).

A formal definition, based on the flatness of the spread-curve, again, requires the ability to tell when the spread reaches the point, where the worm cannot infect "too many" more computers. Depending on the spread technique of the worm, it is possible that the spread curve consists of several growth parts (see Fig. 3b). If we are using a flatness-based definition, we are only interested in the last of these growth parts.

Also, a worm can be designed to have a constant spread-rate of any slope that would allow such a worm to keep just under any predefined flatness threshold, rendering flatness-based saturation definitions useless.

In certain cases we can define saturation in terms of abuse of the major network traffic links, as was the case with SQL/Slammer. Unfortunately, such a definition could only be applied to worms that cause network congestion, making this type of saturation definition non-general.

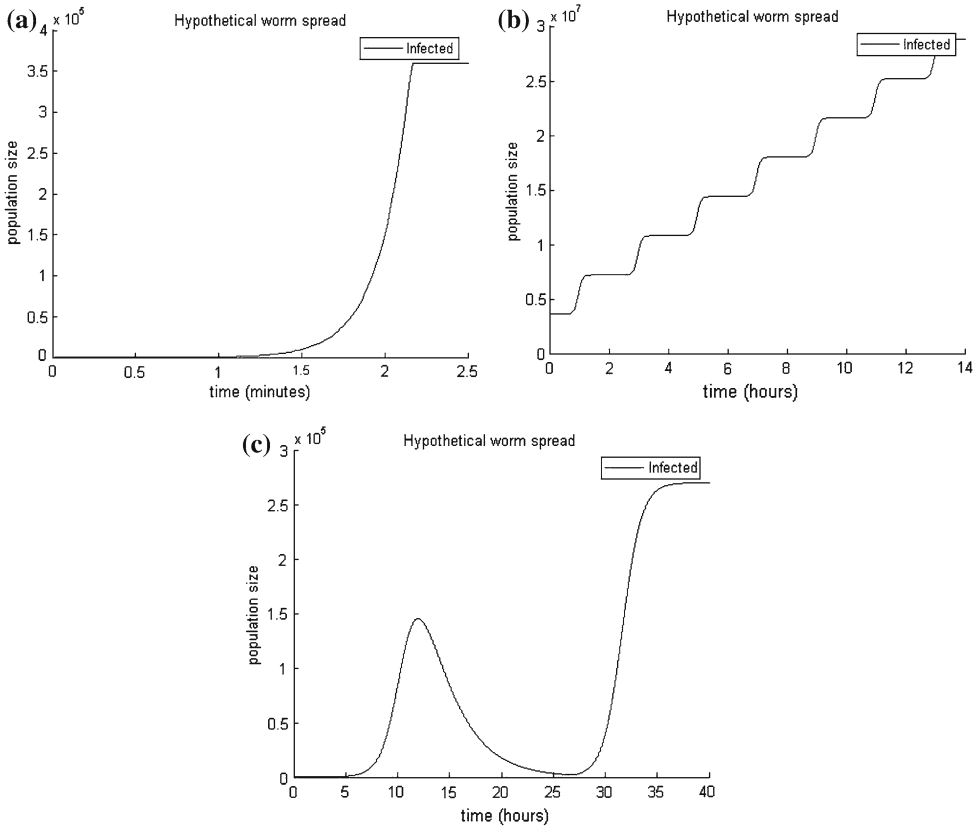

Fig. 3 Possible worm spreads that are contradictory to the general understanding of saturation: **a** a self-stopping worm, **b** periodically restarting spread, **c** a worm overcomes the defense mechanisms

Despite the above mentioned complications, we need to give a formal definition for the term *saturation time*, as it is the base of most of the metrics listed in the previous section. We define saturation time as the time after which the malicious code does not infect new computers.

4 Discussion

It is very important to realize that the application of any of the metrics described above can be perfectly valid if one compares the spread of the same worm under various anti-worm conditions in the environment, as long as the general shape of the spread curve does not change (aside from shifting and multiplication with a constant). The incompatibilities arise when we are trying to apply the same metric to worms with completely different spread characteristics.

We encounter the first incompatibility if we try to compare the sheer number of computers in any stage of two different infections. The number of vulnerable computers is highly dependent on the type of vulnerability the worm is using, therefore the number is likely different for different worms. If we know the exact number of vulnerable machines, it might seem better to compare the percentage of vulnerable hosts the worm infected.

However, using a percentage of vulnerable hosts infected as our metric immediately poses the requirement to know the exact number of vulnerable machines at the point of measurement. This is highly unrealistic for a real-life worm spread, but can be estimated more easily with hindsight, by using the (sometimes grossly incorrect) assumption that the worm could infect each and every vulnerable computer during its lifetime. Furthermore, even if we do know the exact number of vulnerable machines, this number varies so much (based on the fact that different worms use different vulnerabilities), that the metrics can report different results for two (otherwise identical) worms, which exploit similar vulnerabilities in two different software.

However, it is arguable whether the use of percentages instead of raw numbers is preferable. True, it levels the differences between different vulnerabilities, allowing comparison of different spread techniques to see which can contaminate the target population more effectively. However, we argue that in real life a malcode that has spread to more computers causes more damage to the community overall; therefore, the raw numbers are an important factor when measuring the damage the worm causes.

Another problem is *when* to measure the portion of infected computers. Apart from the start, saturation, and the end of the spread, the spread curves can be so different from each other, that specifying a general time that is applicable

to all spreads is impractical. It might seem better to turn the problem upside down and measure either a property of the spread that is independent of the time of measurement, or measure the time itself until a certain (general) condition becomes true.

Metrics presented above that are independent of the time of measurement are *(fraction of) maximum new infections* and *minimum doubling time*. The former is highly sensitive to the length of the sampling intervals, while the later does not tell us when this doubling occurs during the spread of the worm. Arguably, if a worm doubles its infections in the middle of the spread (as in the traditional spread methods), that worm is less dangerous than another that achieves the same minimum doubling time, but doubles its infections throughout its lifetime (as in a conquer-and-divide worm). For an example, see Fig. 3a.

If we are measuring the time until a certain property becomes true, the main question becomes: What property are we interested in? It is important to find a property that happens only once during the spread, otherwise we have to answer how we are going to handle if the property becomes true repeatedly. One way to collapse a series to a single value is to compute some statistics of the series (such as the minimum, maximum, median or mode), and use that single value instead of the whole series. The statistics to use, however, depend highly on the particular property used.

In the paragraphs above, we discussed the potential pitfalls of numerous worm-spread metrics in the case where no counter-measure was considered. In the followings we look at how the situation changes, when we consider the effects of counter-measures on worm-spread.

It is logical that if *all available anti-worm technology together* cannot reduce the number of machines infected by the worm, then they are of no use, even if they slow the spread of the worm down. One could argue that slowing the worm down significantly will allow researchers to find the way to defend against the worm, however, this "new" method should have been considered into the "all available anti-worm technology" part of the previous sentence. While this hypothetical situation is highly unlikely in the real world, it does show that any useful metric has to take into consideration at least the fraction of infectable computers infected. Although it is impossible to take every possible countermeasure into consideration when performing worm-spread simulations, it is sufficient to simulate enough anti-worm methods to show that they are able to reduce the volume of vulnerable machines infected.

5 Proposed metrics

Given the limitations of current metrics, it is clear that measuring the efficacy of anti-worm technologies is non-trivial. Furthermore, in order to provide some measure of "real world" impact, any metric must take into account the *goal* of protection. In terms of a "hierarchy of needs", this protects the network cumulatively before it protects individual systems.

When deciding between the uses of metrics, we need to identify which metric reflects best the damage a worm causes. Damage to the community in the general sense can only be measured in terms of how wide-spread the worm in the network is. Considering this, we chose two metrics: *peak* and *cumulative number of machines infected*. However, in determining these numbers, we must consider the fact that slow moving worms are attacked on a number of different fronts. Thus, our model for prevention must be *systemic*, not local.

These two metrics were chosen based upon two conflicting ways of quantifying damage. First, we consider the damage caused by a worm that rendered infected machines unusable. In this case, the total damage is proportional to the number of machines infected. A second case considers the systemic damage caused by a distributed Denial of Service attack. In this case, the damage scales (though not proportionally) with the largest effective force for causing network disruption (i.e. the maximum number of machines infected at any point in time).

In both cases, we propose comparing solutions by simulating realistic user behavior. For example, if a worm is spreading using a little-known and un-needed feature of IIS, the worm's spread rate will be controlled both by the worm's spread methodology, deployed anti-worm software, possibly network topology, *and the rate at which patches are deployed and new countermeasures are utilized*.

Thus, our goal is to create a virtual test environment where we make assumptions about the speed of reaction of the defenders in terms of remediation. Effectiveness of an anti-worm solution is measured by total number of infected machines and peak number of infected machines during the spread.

The implications of this approach are numerous. First, a protection mechanism that slows the spread of the worm is only meaningful if it slows it sufficiently such that machines are patched/repaired during the outbreak. This might be frustrating to those researchers whose work slows outbreaks but does not prevent them, but it is pragmatically true. Slowing the outbreak is only useful if the slowdown is significant *with respect to other processes in the system*.

Second, this approach recognizes that the total number of nodes infected at any one time and cumulatively is important. The total infection based at any moment is important, as it represents the attacker's largest force multiplier. The cumulative total is also important, as it represents the maximal end-node damage sustained.

The benefit of these metrics is that they measure quantities, which are related directly to the damage potential of the

malcode. However, the drawback is that it is extremely difficult to find the data needed to support the calculations made. However, while this presents difficulties in terms of quantitative results, qualitatively, it should be possible to make reasonable estimates of reaction and patching rates. Further research is needed in this area; however, as currently any model including such factors would be a "best guess" of the actual real-world position.

Even after considering all the above, one could argue that the proposed metrics are too simple and do not include other relevant factors (e.g. targeted attacks, or the cumulative damage caused both locally and systemically). The reason for not including such measurements is threefold. First, it is unclear how much weight each factor should have in the end. If one is interested in protecting the available bandwidth, one would weigh the network traffic more than the number of infected computers; if one is interested in protecting the individual computers, one would weigh the number of infected computers more. Second, and more important, the number of infected computers is the base of all other properties. The amount of traffic generated by the worm can be calculated from the number of infected computers if the spreading technique and exploit used is known.

The third reason for counting only the number of infected computers is that the only common property of malicious mobile code is that they infect computers. Viruses might not use computer networks to spread (earlier viruses spread mainly through floppy disks); contagion worms, for example, multiply "upon request"; and future worms might spread using a medium different from today's network, where the traffic measurement might be incomparable to today's measurements. The only other property that is common in all malicious code spread is the time of the spread, which is only secondary, as shown above, if we also consider the effect of possible defense mechanisms.

5.1 Metrics theory

In the previous section, we argued that our two chosen metrics are the best measurement of the damage a malicious mobile code causes to society on a holistic level. However, before we can say that our metrics are the right ones, first, we have to show that the proposed measurement techniques are indeed metrics. We will find interesting conclusions.

The theory of metrics says that the thing we want to measure (in our case, damage to society) might not be captured precisely, but certain directly associated and measurable *factors* can give a clue into our object of interest. The factors can be thought of as attributes that provide a direct measure on the object of interest. Note here that the factors also must be measured. The vicious cycle is broken by assuming that there is enough consensus between the experts of the field that the chosen feature represents the object of interest.

Once we have a feature, we can look at certain elements of the system and apply a function on them to yield a number. The function applied is called the *metric*. We first have to *validate* our metric, by taking a system, applying the metric, and comparing it to the measured factor. Once the metric is validated, we can apply it to other systems (and possibly re-validate it to increase our confidence).

It is worth noting that it is always preferable to use the factor, if possible, to relate to the object of interest. However, most of the time, we do not have access to the factor, but still would like to get an estimation of the object of interest. In a concrete example (taken from [6]), our systems could be software projects, the object of interest can be the quality of the software, and the chosen factor can be number of errors found in the software. Obviously, early in the development phase we do not have the software yet, and so we cannot have access to the error count (our factor) either. Here is when metrics come into play. We can apply a function on some elements of the code (e.g. lines of code or vertices and edges in the control graph representation of the code) to get a *prediction* on the error count. Other uses of metrics are *assessment* and *control*, but not every metric is suitable for each use.

When we are drawing an analogy between the above software metric example and our case of malicious mobile code, some translations are easily made: our system is the network, in which the malcode spreads; the object of interest is the damage, caused by the worm, on society as a whole; as we argued in earlier sections, the factor is the number of infected computers (peak or cumulative). However, what should be the *metric*, which must be a function, and what are its parameters?

The answer is that the metric function is the model itself. We can apply an analytical model, a Monte-Carlo simulation, or any other way that computes the number of infections (including a "live" experiment on the Internet, although that will have some ethical issues) from its input parameters (see Fig. 4). In simulations (Monte-Carlo and test-bed) the interesting thing is that some of the parameters cannot

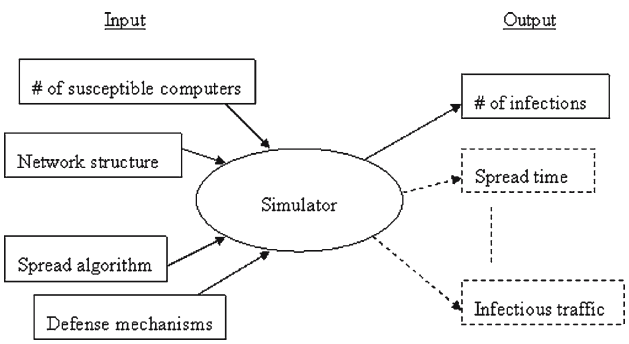

Fig. 4 Application of a simulator as metric on the spread of a malcode

be described by numbers, as they are algorithms (e.g. target selection of the worm, traffic on the network, behavior of the countermeasures).

Therefore, if we want to be precise, the number of infections when measured on the real-life spread of a worm is the factor (an attribute of the damage), and the metric to be applied is a particular model that will also yield the number of infections. Validation of a metric (model) can then be the comparison of the yielded number and the observed number. Looking from this perspective, however, it is obvious, that comparing only the (peak or cumulative total) number of infections is not sufficient to validate a model (the parameters can easily be tweaked to produce the desired result). Instead, the validation needs to be a fitting of the curves, using some statistical techniques in case of methods that have internal variations (Monte-Carlo and test-bed experiments). This can be viewed as a multiple validation at different points in time during the spread of the worm.

Once the necessary validation is performed, the model can also be used to predict the efficiency of worms and countermeasures in different hypothetical scenarios.

6 Conclusion and Future Work

The science of measuring the efficacy of anti-malware techniques is still in its infancy. Furthermore, there has been little work placed upon the important problem of measuring the relative effectiveness of different anti-worm techniques.

In this paper, we have outlined several of the "obvious" spread metrics and shown how they can lead to misleading results. We therefore described two metrics which can be used to make meaningful measures of worm suppression in terms of real-world damage and overall spread. These metrics can be used to compare different approaches, and allow both defenders and researchers to model the effects—both locally and globally—of different protection mechanisms and reaction times.

The methods described here need significant further work in order to be effective. In particular, realistic models of patching, firewalls and response time need to be created. However, the lack of these models should not be seen as a fatal flaw: in order to measure and model malware spread, such details *must* be included in any complete calculation.

Our Monte-Carlo approach has shown great promise in this area. In earlier research, our models reproduced the spread of various well-known worms with high reliability. These results provide us with some level of confidence that the system can be used predictively, in order to model protection techniques before they are deployed in the real-world, and, crucially, provide metrics by which different protection scenarios can be modeled.

References

1. Filiol, E.: Computer viruses: from theory to applications. Springer, France, ISBN 10: 2-287-23939-1 (2005)
2. Gordon, S.: The generic virus writer. In: Ford, R. (ed.), 4th International Virus Bulletin Conference. Jersey, UK (1994)
3. Gordon, S.: The generic virus writer II. In: Proceedings of 6th International Virus Bulletin Conference. Brighton, UK (1996)
4. Kim, H-A., Karp, B.: Autograph: toward automated, distributed worm signature detection. In: Proceedings of the 13th USENIX Security Symposium. San Diego, CA (1996)
5. Newsome, J., Karp, B., Song, D.: Polygraph: automatically generating signatures for polymorphic worms. In: Proceedings of the IEEE Symposium on Security and Privacy. Oakland, CA (2005)
6. Schneidewind, N.F.: Methodology for validating software metrics. IEEE Trans. Softw. Eng. **18**(5), 410–422 (1992)
7. Staniford, S., Moore, D., Paxson, V.: How to own the Internet in your spare time. In: Proceedings of the 11th Usenix Security Symposium. San Francisco, CA (2002)
8. Wagner, A., Dübendorfer, T., Plattner, B., Hiestand, R.: Experiences with worm propagation simulations. In: Proceedings of the 2003 ACM workshop on Rapid Malcode (WORM). Washington, DC (2003)
9. Wang, Y., Wang, C.: Modeling the effects of timing parameters on virus propagation. In: Proceedings of the 2003 ACM workshop on Rapid Malcode (WORM). Washington, DC (2003)
10. Weaver, N., Staniford, S., Paxson, V.: Very fast containment of scanning worms. In: Proceedings of the 13th USENIX Security Symposium. San Diego, CA (2004)
11. Williamson, M.M.: Throttling viruses: restricting propagation to defeat malicious mobile code. In: Proceedings of the 18th Annual Computer Security Applications Conference. Las Vegas, Nevada (2002)
12. Williamson, M.M., Lévillé, J.: An epidemiological model of virus spread and cleanup. In: Proceedings of the International Virus Bulletin Conference. Toronto, Canada (2003)
13. Zou, C.C., Gong, W., Towsley, D.: Worm propagation modeling and analysis under dynamic quarantine defense. In: Proceedings of the 2003 ACM workshop on Rapid malcode (WORM). Washington, DC (2003)