A Study on Friends Model of a Computer Worm Defense System

Rashid Husain, Mansir Abubakar

***Abstract*— This paper addresses the problem of computer worms in the modern Internet. A worm is a self-propagating computer program that is being increasingly and widely used to attack the Internet. This research paper develops a friend model of a computer worm and discusses in length the aspects involved in defending the Internet against a worm. Of primary interest are models that can automatically respond to a worm outbreak. It discusses the results of real time experiments conducted on the campus gateway for the `Trend Center' effort and the results of simulations of the mitigation models. It concludes that worms are dangerous to the Internet but there are ways and means to mitigate their ill-effects.**

***Index Terms*— Worm, Friend Model, Internet, Hosts, Infection, Protected**

I. INTRODUCTION

This model of defense is based on the willing co-operation of a set of hosts on a pre-arranged protocol which is described below. Once a worm is detected, an alert message is spread to the set of participating hosts to stop the spread of the worm [1]. This alert can be sent from the detector to the entire set or a small subset of participating hosts depending on the detector's ability to reach other hosts and other factors discussed below. Our goal is to maximize the number of hosts that can be prevented from contracting the worm.

We develop mathematical models for the simplest of the scenarios. Then, we go on to develop simulations to study more complex scenarios of worm mitigation [2,3].

II. THE MODEL

2.1 Assumption

We assume the following for the sake of simplicity:

- The worm is not polymorphic, and its signature is known in advance or it is identified in real time.

2.2 The Worm Race

All participating hosts have a peer-to-peer and a hierarchical relationship with other hosts as in the real Internet. For eg., Border Gateways are higher in the hierarchy than routers while all routers and all B_{GWS} have a peer-to-peer relationship amongst themselves respectively [3].

Each participating host has a list of trusted hosts1 with associated trust-worthiness. By default, the trust is relative to the position in the hierarchy. The trust-worthiness of a host higher in the hierarchy is higher than that of a peer. This list need not include all the participants.

Once a host suspects a worm, it sends the suspected signature and an associated perceived threat, *PT*, to its friendly or trusted hosts. Such alerts received the trust-worthiness of the alerting hosts and the actual numbers of suspicious packets seen by the host form the inputs to calculate *PT* [4, 5].

Perceived Threat, *PT* = f (number of alerts received, trust-worthiness of the alerting hosts, number of malicious packets seen) …… (A)

f is a function that is decided by the participant to calculate PT. If the PT exceeds a certain pre-determined threshold and if any other criteria that the host has determined should be met, are satisfied, then that host takes actions. An example of a criteria that a host can require to be met is that any alert must be received from a certain number of different sources, each of which have a certain level of trust relationship with this host [18]. This reduces the number of spurious alerts originating from malicious or already compromised hosts.

The list of friends is not static. It can be changed on a periodic basis based on the veracity of the previous alerts from each of them automatically or by the site administrator [17]. The protocol to be followed while updating the friends lists will be the same as in updating routing tables to avoid race conditions and other inconsistencies that are possible when data at different locations have to be updated independently. Of particular interest would be the situation where one or more of the participants are left out of the friends lists of all the other participants due to race conditions. This situation should be avoided.

The actions taken are decided by the hosts individually unless there is an understanding with other hosts to take a collective decision about the actions to be undertaken [13,15].

The iterative actions taken by each host include:

1. Assigning a threat level as perceived by the alert-receiver based on various parameters as mentioned above. This PT may be different from the PT seen by others.
2. Alerting its friends. Depending on the PT, a host chooses a different number of friends to alert.
3. Scanning the incoming and outgoing packets for the new signature . The intensity of scanning depends on the PT at the host. Based on the results of scanning, the PT at that host might increase or decrease. The intensity of scanning is the sampling rate of the traffic.
4. If the PT changes based on scanning, sending new alerts with the new PT. This acts like a control mechanism to dynamically increase or decrease the scanning intensity.
5. Reducing the bandwidth available to the general traffic and increasing the bandwidth to the alert messages. This prevents the worm traffic to occupy all available bandwidth that would prevent alert messages from propagating. This is possible because the hosts can control the traffic speed passing through it.
6. Blocking of traffic that is believed to be malicious.
7. Backing off from blocking the allegedly malicious traffic once it is found not to be so. This is achieved automatically

Rashid Husain, Department of Mathematics and Computer Science Umaru Musa Yar'aduva University, Katsina, Nigeria

Mansir Abubakar, Department of Mathematical Sciences Al – Qalam University, Katsina, Nigeria

since the intensity of the scanning decreases if there is a decrease in PT.

This spread of the alert messages may also be seen as a worm, but benign. Hence the topic Worm Race. One should also note that since we, the defenders, know the targets for our alerts, we can spread the benign worm much faster than the malicious worm after the initial latency of detecting a worm and extracting its signature [9].

The defending hosts might include backbone switches, routers, border gateways or gateways at universities. The higher in the network hierarchy that we deploy our scanning; the more costly is the process because of the high volume of traffic. But we can stop the worm at a much earlier stage and can also spread the alert message quickly as there are fewer targets for our alerts.

Note that if the worm has already crossed a host, a router for example, and entered the network below it, the hosts in the sub-net are open to attack. So, we need to have the detectors at various levels of the Internet hierarchy, not just at one level. For example, both at border gateways and routers; not just at border gateways. We should also not increase the redundancy too much lest we inflict a self Dos where a majority of the machines end up just scanning the traffic passing through it. We need to choose an optimal redundancy level to avoid such single point failures [7,8].

Currently this model proposes scanning at 2 levels, one at the B_{GWS} level and the other at the routers, one level of hierarchy below B_{GWS}. Fig.2.1 shows the hierarchical relationship among the B_{GWS}, routers and hosts.

While scanning incoming traffic prevents infection, scanning of out-going traffic prevents the spread of the worm and helps in quarantining the infected sub-net [10].

III. MATHEMATICAL MODELS

Mathematical models for the spread of worms are very similar to the epidemiological spread of diseases. In both the cases, the rate of spread of victims is proportional to both the number of victims and the number of susceptible hosts available for further infection [12]. Thus, if B a is the fraction of infected hosts, the rate of spread of infection is given by:

$$\frac{da}{dt} \propto a(1-a)$$

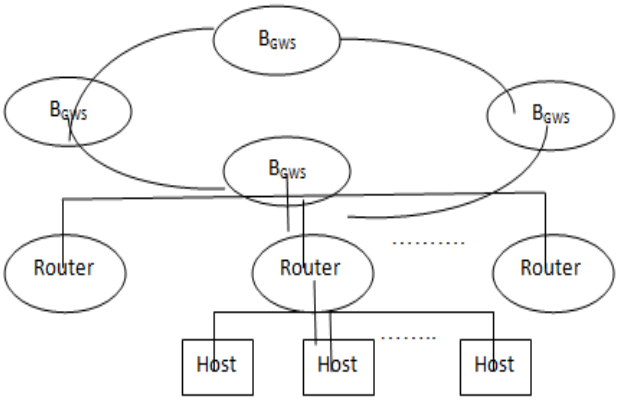

Figure 2.1: Hierarchical relationship among B_{GWS}, Routers and Hosts. Ordinary hosts are one level below routers which are one level below B_{GWS}.

Introducing a constant of proportionality, K, which here is the same as the rate at which each infected host can find and infect other hosts, we get [19, 20],

$$\frac{da}{dt} = ka(1-a) \quad ----- (2.2)$$

Solving this differential equation, we get,

$$a = \frac{e^{k(t-T)}}{1 + e^{k(t-T)}}$$

where T is a constant that fixes the time position of the incident. The equation produces the S-curve growth behavior typically seen in population growth models and is known as the logistic equation. This is the same equation that governs the rate of growth of epidemics in finite systems entities.

This is an interesting equation. For early t(t << T), a grows exponentially. For large t(t << T), a goes to 1 (all vulnerable hosts are compromised). This is interesting because, it tells us that a worm like this can compromise all vulnerable machines on the Internet quickly [22].

Mathematics of the Friends' Model The rate of spread of worm for the Friends' model described in this paper has been developed in [14]. It is presented here in detail for the sake of completeness. The variables are defined as follows:

M : the total number of response members (eg. routers)

a : the number of infected members

c : the proportion of alerted members

F : the number of friends for each host

α : the function of alerts a host needs before it crosses the threshold

σ: the severity of the alert

For a given member, the expected number of co-operating friends who remain in the normal, un-alerted state is:

$$F.(1-c)$$

In our simple strategy, a cooperating member increases the severity of an alert in proportion to the number of infection attempts that it sees [8]. Therefore, the number of alerts a particular member sends in time dt is:

$$F.(1-c).\sigma\alpha.dt$$

This implies that the total number of alerts system wide is given by multiplying the above term by M, the total number of response members. Since each member needs α alerts before it can change its state, the number of members changing state in time dt is given by:

$$\frac{dcM}{dt} = \frac{F.(1-c).\sigma\alpha}{\alpha}$$

Rearranging the terms, we get the evolution rate of the number of alerted members in the following differential equation:

$$\frac{dc}{dt} = \frac{F.(1-c)}{\alpha}.\sigma\alpha \quad ------(2.3)$$

The proportion of member already infected is obtained by altering (2.2) to include the fact that cooperatively alerted members will be able to block the worm. Two types of infection attempts are considered, local and global infection. Local infection is an infection that spreads from a host to another host without having to pass through any router [6, 11]. Global infection is an infection that needs to pass through a router. When an infected host tries to infect another one across a router, the infection must pass through two filters, the local filter that blocks outgoing infections and the remote filter that blocks incoming infections.

The probability that both of these are not alerted is $(1-c)^2$. Thus the global infection is

$$ak(1-a)(1-c)^2$$

The local infection rate is same as the rate equation (2.2) because there are response devices between infection source and target. Since there are M response members, the probability of a host choosing a target behind the same router is 1/M and behind another router is 1-1/M. Combining these probabilities and the infection rates, equation (2.2) becomes:

$$\frac{da}{dt} = ak(1-a)(1-c)^2.\left(1-\frac{1}{m}\right) + ak(1-a).\frac{1}{m} \quad ------(2.4)$$

Thus we have a pair of differential equations which can be solved to get the number of infected and number of alerted members.

Dynamic response strategy with back off mechanism

According to our model, hosts back off from filtering the traffic after a certain time period. The rate of back off is inversely proportional to PT meaning, it is directly proportional to (1 - a) and also to the proportion of alerted members, c. Thus equation 2.3 becomes [21, 22]

$$\frac{dc}{dt} = \frac{F.(1-c).\sigma a}{\alpha} - \in (1-a)c \quad ------(2$$

where $\in$ is a constant indicating how fast a host backs-off from filtering traffic [14].

IV. DESCRIPTION OF THE SIMULATION

The worm defense model was simulated using the Swarm simulator. Swarm is a software package for multi-agent simulation of complex systems like population dynamics and simple social behavior in biological organisms [2]. Swarm was chosen for its ease of programming and its active developmental support.

We consider a rectangular grid of x by y routers each having 8 neighboring hosts connected to it. There are two kinds of routers and hosts. One kind that is infected with a worm and the other alerted by the white worm [6,7].

The simulation begins by starting the infection from random hosts. An arbitrary host is then marked as the initial detector of the infection. This host sends an alert to 8 of its friends. These friends are chosen randomly for the purposes of this simulation. The alerted hosts then calculate the threat that they perceive, PT, according to equation. 2.1. If the PT that each of these friends calculate exceeds a threshold and if certain conditions that are required to be satisfied by that host are met, then each of them also takes actions which may include alerting their friends in turn [13,14]. Thus the alert message propagates from one host to another. This alert, the white worm spreads until all routers are alerted. Once all routers are alerted and start scanning traffic, the worm can no longer spread, thereby trapping the worm within those domains which are already infected. Once the worm stops spreading, the routers in whose sub-net there is no infection back off from scanning the traffic, while routers with infected sub-net continue to scan traffic. This reduces the number of routers that scan the traffic to the minimum required to stop the worm spread. This reduces the cost of blocking worms to the minimum required [7].

V. DISCUSSION OF THE RESULTS

The curves for a typical scenario with 8 friends for each host are shown in Fig. 2.2. Initially when the worm starts spreading, none of the hosts are aware of it. But once one host raises an alarm, a large number of the participants are alerted in a cascading fashion and traffic is monitored [13,17]. The graph shows that the alert messages spread much faster than the worm and thereby triggers the filter rules at a large number of places before the worm could reach them. We can see that the maximum infection is restricted within 25% of the population. The drop in the infected population is due to the patching of machines against the worm. We can change various parameters such as the number of friends each host has, the spread speed of the worm, the network scanning technique the worm uses, etc., in the simulation to study the effectiveness of our model.

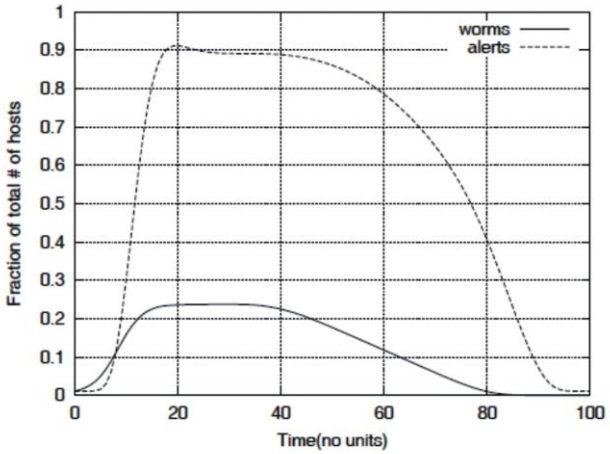

Figure 2.2: The graph shows how the alert messages catch up with the worm.

VI. LIMITATIONS

This model could fail if the network is already saturated with malicious worm traffic. Reference [7] deals with such network connectivity problems. The techniques described in [7] can be used to analyze the robustness of network architectures against denial-of-service by link and node destruction.

If we treat our friends' network as a graph with hosts representing nodes and the links between them as edges, we could answer the question, `how many links may get saturated before the alert messages can't reach others?'

N : Total number of participating hosts

F : Number of friends for each host

T: Number of independent hosts from which alerts are required before any action can be taken

There are F unsaturated links from each host to its friends when there is no worm. If this host has to take action, it should have at least T links unsaturated to receive T alerts. Thus, a maximum of $F - \mathrm{T}$ links could be saturated. If more than $F - \mathrm{T}$ nodes are saturated, then this particular node cannot receive the required number of alerts to take any action.

Considering the links to be uni-directional, since there are N hosts in total, we could have a maximum of $N.(F - \mathrm{T})$ links saturated before the model can fail if failure is defines as any host has more than $F - \mathrm{T}$ links saturated. Beyond that

for each additional link that gets saturated, a maximum of one node gets cut-out from the network. In the worst case, if each additional link beyond $N.(F-\mathrm{T})$ that becomes saturated cuts o_ one node all nodes are isolated when $N.(F-\mathrm{T})+N$ links get saturated [20,21,22].

Another limitation of this model is that the signature of the worm is assumed to be available already or almost immediately after the worm is detected. But, it is very difficult to get the signatures of zero-day attacks within a very short time. Because, there are several techniques for very high-speed worms, this model could come a cropper against them.

VII. CONCLUSION

This model uses a peer-to-peer strategy to control large scale worm attacks on the Internet. It shows that a controlled white worm propagating faster than the worm is an effective strategy to minimize the number of worm victims. This chapter provided a mathematical model for the spread of worms in general. It also gave expressions for the rate of spread of the infection as well as the white worm. Simulations with smaller number of friends for each node suggest that a larger number of friends would suppress worms better. But simulations also showed that the number of alerts exchanged grows exponentially and the network performance degrades. Also, it is not scalable to the real world as one cannot have a large list of trusted friends. A large number of friends tends to give rise to a large number of false alarms. A certain amount of false alarms is inevitable. But the system suffers when faced with a large number of false alarms. However, when we use an optimized friends list this model is able to contain the worm to fewer hosts. In the simulation environment which had 5761 routers in a grid of 81 X 81 with 8 hosts connected to each router the optimized list turned out to have 8 friends.

REFERENCES:

[1] C.G.Senthilkumar, (2002). "Worms: How to stop them? - A proposal for Master's thesis," *University of California, Davis. http://wwwcsif.cs.ucdavis*.

[2] C.G.Senthilkumar and Karl Levitt, (2003). "\Hierarchically Controlled Co-operative Response Strategies for Internet Scale Attacks," *University of California, Davis. http://wwwcsif.cs.ucdavis.edu*.

[3] D.Farmer and W.Venema, (2004). "Security Administrator's tool for analyzing networks" *http://www._sh.com//zen/satan/satan.html*.

[4] D.Noijiri, J.Rowe, and K.Levitt, (2002). "Cooperative Response Strategies for Large Sacle Attack Mitigation," *DISCEX*.

[5] Mark W. Eichin and Jon A. Rochlis, (1988). "With Microscope and Tweezers: An analysis of the Internet Virus of November 1988," *In Proceedings of the symposium on Research in Security and Privacy, Oakland, CA*.

[6] Dan Farmer and Eugene H. Spa_ord, (1990). "The cops Security Checker System ," *USENIX*.

[7] Gene Kim and Eugene H. Spa_ord, (1993). "The design of a system integrity monitor: Tripwire," *Technical Report CSD-TR-93-071, Purdue University, West Lafayette, IN, USA*.

[8] David Moore et al. (2003). "Inside the Slammer Worm," *In IEEE Security and Privacy*.

[9] Carey Nachenberg, "Computer Parasitology," *Symantec AntiVirus Research Center*.

[10] Carey Nachenberg. "Understanding and Managing Polymorphic Viruses," *Symantec AntiVirus Research Center*.

[11] Don Seeley, (1989). "A Tour of the Worm," *In Proceedings of 1989 Winter USENIX Conference*, pp. 287 -304.

[12] John F. Shoch and Jon A. Hupp, (1982). "The Worm Programs - Early Experience with a Distributed Computation," *Communications of the ACM*, Vol.25(3) pp: 172 -180.

[13] Eugene H. Spa_ord, (1988). "The Internet Worm Program: An Analysis," *Technical Report CSD-TR-823, Purdue University, West Lafayette, IN, USA*.

[14] Eugene H. Spa_ord, (1989). "The Internet Worm: Crisis and aftermath," *Communications of the ACM*, Vol. 32(6), pp: 678 – 687.

[15] Eugene H. Spa_ord, (1991). "The Internet Worm Incident," *Technical Report CSD-TR-933, Purdue University, West Lafayette, IN, USA*.

[16] S.Staniford-Chen et al. (1996). "A Graph-Based Intrusion Detection System for Large Networks," *In The 19th National Information Systems Security Conference*, Volume 2, pages 361 – 370.

[17] Stuart Staniford, Gary Grim, and Roelof Jonkman, (2001). "Flash Worms: Thirty Seconds to Infect the Internet," *Silion Defense - Security Information*.

[18] Stuart Staniford, Vern Paxson, and Nicholas Weaver (2002). "How to Own the Internet in Your Spare Time," *In Proceedings of USENIX Conference, Berkeley, Usenix Association, USENIX*.

[19] Nicholas Weaver, (2002). "Future Defenses: Technologies to Stop the Unknown Attack Internet", *http://online.securityfocus.com/infocus/1547*.

[20] Nicholas Weaver, (2002). "Potential Strategies for High Speed Active Worms: A Worst Case Analysis," *UC Berkeley*.

[21] Nicholas Weaver, (2002). "Warhol Worms: The Potential for Very Fast Internet Plagues," *UC Berkeley*.

[22] Tarkan Yetiser, (1993). "Polymorphic Viruses Implementation, Detection and Protection," *VDS Advanced Research Group*.

[23] Dan Zerkle and Karl Levitt, (1996). "Netkuang - a multi-host con_guration vulnerability checker," *USENIX*.