This is to certify that the
thesis entitled

PROTECTING NETWORKS FROM SINGLE-PACKET WORMS

presented by

LARRY GRANGER IRWIN II

has been accepted towards fulfillment
of the requirements for the

M.S. degree in Computer Science

Major Professor's Signature

7/11/05

Date

PROTECTING NETWORKS FROM SINGLE-PACKET WORMS

By

Larry Granger Irwin II

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Computer Science

2005

ABSTRACT

PROTECTING NETWORKS FROM SINGLE-PACKET WORMS

By

Larry Granger Irwin II

Can a network be protected from single-packet Warhol worms [14]? This paper generates and simulates random network environments to answer that question. The research assumes a perfect detection algorithm and varies the time required to perform the identification. Perfect detection alone is not sufficient; it must also be swift in recognizing threats as some cases presented here show that perfect detection offers no noticeable protection. The impact of other network factors on worm propagation and prevention are investigated as well, including: router participation in the prevention scheme, the percentage of routers involved in the traffic passing, and the ability for participating routers to communicate. The results are promising: realistic simulations without communication can protect over 50% of the network. The addition of communication increases that protection to over 80%. The key result is that emerging identification technologies such as LeBrea [31] can be leveraged into viable automated network protection systems against single-packet worms.

ACKNOWLEDGMENTS

Many thanks to the members of my graduate thesis committee: Dr. Richard Enbody, Dr. William Punch, and Dr. Abdul-Hossein Esfahanian for their assistance and insights. Thanks also to John Kyritses and Scott Wainstock for their time and opinions. Much appreciation also goes to the Davison public library of Davison, Michigan, for providing a quiet workspace. And, finally, I thank my fiancée, Katie, who slept alone too many nights for academia's sake.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 - INTRODUCTION

Self-propagating code, more commonly referred to as a "worm", is a computer program engineered to place copies of itself on a remote machine without the express consent of any of the machines involved. The specifics of how a worm works is outside the scope of this document and the interested reader is directed to [2, 4, 13] for further information. Programs of this nature have existed publicly as far back as 1988 [4, 13]; however, worm epidemics have recently grown in both scope and severity [14].

Factors contributing to the rise in worm activity range from the number of devices attached to the Internet, to the skill of those who author the worm software. Further, despite the fact that many of the more prolific worms have been engineered to be relatively harmless; there is no definitive means to stop someone from creating and releasing malicious self-propagating code into the wild. Finally, it has been theorized that a worst-case worm could reasonably cause upwards of US$50 million dollars as a result of various types of damage and downtime [32]. It is paramount that effective defense mechanisms be deployed to limit the severity of said "worst-case worms". This paper examines and devises a first generation defense system that is capable of providing automatic protection from the spread of a single packet worm.

There have been multiple strategies investigated as possible methods for combating the spread of worms. These methods include address blacklisting and content filtering [9], selective shutdown [10] and traffic throttling [11] (which at one time had been planned for commercial release).

Address blacklisting [9] involves recording the IP address of a machine that is believed infected and disallowing any traffic from that machine to be forwarded.

Selective shutdown [10] takes the removal of the machine one step further and actually initiates a power-down procedure for machines that are infected. These two approaches, if implemented as detailed in the literature, would reduce the throughput of the machines to zero. Subsequently, this paper investigates possible solutions involving content filtering and throttling [9, 11]; two approaches that allow the compromised machine to be used despite it succumbing to the worm.

The difficulties associated with constructing physical labs to conduct experiments on a grand scale (as well as the threat of releasing malicious code into the wild) have forced many researchers to rely on various simulation models to conduct experiments. Chen and Robert discuss how the simple epidemic model taken from biological epidemiology is a good estimate to the behavior of worms in high-speed networks [2]. Further, they take the results of the mathematical models to underscore the importance of real-time detection and accuracy that would be required for any successful prevention scheme. Weaver, et al. uses a similar mathematical models and an abstract simulator to demonstrate how worms could theoretically reach astronomical infection rates [14]. They describe two types of potentially harmful worms: the Warhol or flash worm and the surreptitious worm. These are worms that threaten the health of networks by either spreading so fast that no counter-measures are futile, or spreading so slow as to make detection practically impossible, respectfully. The single-packet Slammer worm is used as real life evidence regarding the speeds that worms may exhibit currently as well as possible speeds that could be reached in the future. Again, Weaver and company argues for a central defense organization likened to a digital center for disease control.

Others have focused on the simulation and examination of worms that were successful in reality. Moore, et al. take an in-depth study of the Sapphire/Slammer worm in [27]; discussing its strengths and weaknesses while emphasizing that no human administrators could possibly have prevented the outbreak. Chen, Gao and Kwait extend the simple epidemic model in [26] to account for observed behaviors in reality that does not coincide with established models. Code Red v2 is simulated using the new model and results are offered that more closely resemble those observed in reality than that which has been produced by previous models. The model is finally used as a tool to evaluate the effectiveness of the LeBrea defense system [31] on various levels of monitored address space. Zou et al. also investigate the Code Red worm and its variants in [25] and develop their own model to account for the observed behavior. The authors of [25] dispute rather accepted values regarding the overall infection of Code Red and use a connected model such that "there is no topology issue in our simulator". Despite the lack of topology, the model produces results that are similar to other models and can account for various factors that cause a drop in infection during the final stages of the worm's progression.

There is also another set of work from those who have examined possible counter-measures to defend against the spread of malicious self-propagating code. Wang, Knight and Elder explore the effect of immunization on various topologies in [29]. The work was to simulate the environments of large organizations and immune nodes would not allow infections to pass through. Thus all nodes acted as routers; unfortunately, the hierarchical model was a tree of nodes and thus all non-leaf nodes were cut-nodes and immunization effectively segmented the network. However, the work utilized a count of infection attempts to gain a measure of the efficiency of the worm as well as gain an idea

as to which portions of the network were most valuable in protecting. Perhaps the work most similar to that, which is presented here, is the work of Moore, Shannon, Voelkner and Savage [9]. The aforementioned authors utilize mathematical and proprietary simulations to investigate the effectiveness of containment strategies, reaction time and blocking location. The work concludes with recommendations regarding each of those three aspects and warns that creating such an automated system will be "very challenging". The work presented in this research differs from [9] in many ways: the simulation software is open-source, the networks are randomly generated, and the percentage of the network that is utilized in packet forwarding is a parameter to the simulation.

The complete spectrum of worm variants and techniques are vast and consequently are outside the scope of this research. This paper selects to focus on one particular type of worm: a single-packet Warhol type worm as described in [14]. Single-packet worms are limited only by the available bandwidth since no data management overhead is needed because the exploit only requires one data packet. The simulation software is also custom-built as well with the specifics of this research in mind. This custom software is not a unique occurrence as none of the previously mentioned research utilizes identical frameworks.

The remainder of this paper is organized as follows. First, section 2 describes in detail how the simulation is structured and operates. It details the logical and physical components of a network and how they are represented in this simulation software. A base case using this simulation is compared against published accounts of worm propagation to support the simulation's accuracy. Then, Section 3 describes the results

for the various traffic throttling and content filtering experiments using various levels of routers participating as reactionary routers. Section 3 also introduces the difference between the realistic and extreme simulation parameters and examines various types of methods to distribute knowledge amongst the reacting routers. Section 4 summarizes the results obtained from the experimentation and explains the observed behavior. Finally, Section 5 discusses some weaknesses of the current software and examines some topics and extensions to this paper that the author believes worthy of additional attention.

CHAPTER 2 - SIMULATION SOFTWARE

Various methods exist that allow one to simulate the behavior of a network as well as worms. These simulation packages include NS2 [22], SSFNet [23] and various academic packages [24, 28]. All simulations are close approximations to the true behavior of the actual Internet, but the web's large size and dynamic nature make it an open question as to how to correctly simulate the environment. Worm simulations are debated as well [25,26,27] with models ranging from the biological epidemic model [2] based on how disease spreads within a population to the two-factor worm model [25] where technical aspects are considered including link utilization, system death and patching. However, these simulations are not easily modified to account for routers that have varying behavior; a modification to alter the basic elements of NS2 and SSFNet appeared to require intimate knowledge of the simulation and as such was not a viable option for this research. Other models were mathematical and would have been simpler to include modified behaviors, however the existing model was without any topology [25]. Therefore, the existing research serves as a guide and basis on which to judge the accuracy of the simulation developed for this research.

2.1 Implementation

The simulation software is completely written in C++. The source compiles under both Visual Studio 2003 and GNU gcc version 3.4.1.

The system is comprised of four main components intended to represent basic elements of a network: autonomous system (AS), router, node and packet. An AS is defined as a collection of IP networks under control of a single entity, typically and Internet service provider [30]. Each component is an object and the relationships between

these objects as well as their interactions are described in more detail within the following sections.

2.2 Design

The network simulator was designed to accurately model the majority of Internet traffic, yet be simple enough to allow for many changes with little effort. Such a simulation is far from trivial [21] and the current implementation makes no claims to be an exact simulation engine. The software has been named NetSim and that name will be used within this paper to refer to the software utilized for this research. The pseudo networks within a NetSim simulation are based on the various tiers that exist in the current Internet and are internally represented by a tree of autonomous system objects (AS), similar to the topology implemented in [29]. The specific parameters that determine the shape of the tree are defined by a set of simulation constants. This structure is influenced by the current general construction of the commercial Internet: tier-1 Internet service providers (ISPs) that lease access to regional ISPs, which in turn further offer access to local ISP providers [7]. Instances do arise in the Internet where ISPs (which one may also consider an AS in this context) develop an agreement with another ISP that it is not directly connected with to set up a link to allow each other's traffic to pass through. These links (sometimes referred to as peer connections) are not currently included in the model is one area of improvement listed in Section 5. Such a tree of AS objects may be used to simulate the Internet because of the "significant degree of hierarchy" [20]. Furthermore, others have used similar techniques of using trees to model the behavior of large networks [19].

Anecdotal evidence towards the validity of the structure used in NetSim comes from *traceroute*. When one performs a *traceroute* to various locations around the country (or the world, in most cases) the beginning of the route will almost always contain many of the same hops. This consistency in hops is a result of the tier structure and the relatively low frequency of peer agreements [7]. Despite whatever downfalls that may exist in the design and implementation of the network, the pitfalls will equally affect the worm as well as the countermeasures that are examined. It is the hope of the author to create a virtual environment close enough to warrant a more thorough and exhaustive examination of the theories introduced here. The simulation paradigm implemented is a step-based simulation. There is no direct correlation between steps and time to allow for the timing and experiments to be abstracted out in relation to the amount of steps needed for certain operations to be completed. This will allow for the response time for the routers to be directly scaled to the speed at which a simulated worm propagates.

2.2.1 Physical Entities

The basic organization of objects that represent the physical components of the pseudo-network is a tree at the highest level of abstraction (see Figure 1). Notice that it is possible to set up the parameters such that an AS object generates zero child objects. The tree starts with the root AS and then a random number of child objects are added to each AS in a depth-first traversal until a maximum threshold is reached. This threshold could relate to the total number of AS, node or router objects created thus far, or the depth of the AS objects within the network structure. All of these simulation parameters are located in a constants file and appear in Appendix A.

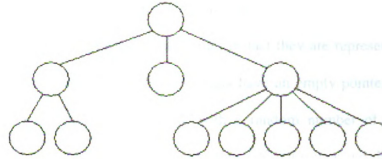

Figure 1: Example AS Tree Structure The above k-ary tree is an example of how a collection of autonomous systems may be arranged internally to represent a virtual network.

2.2.1.1 AS Description

Each AS object has a randomly generated set of children as long as certain thresholds regarding the number of existing AS objects, routers, nodes and current AS depth have not been met. A random number of routers are created within each AS object. Furthermore, there are two different types of routers that exist within an AS: internal routers and gateway routers (for a detailed description, see Router Description). Each set of routers is randomly generated in separate steps during network creation. Two pairs of simulation constants limit how many of both types of routers are generated.

These constants that restrict router creation are not always true constant values. Another simulation constant determines if the pair of aforementioned constants act as definite values or definite factors. For the purpose of this research, the constants have been arranged such that the restrictions on routers are based on the current depth of the AS within the network simulation tree. These factors allow for more routers and fewer nodes in the upper level AS objects; vice versa as the AS objects are placed lower in the tree.

2.2.1.2 Router Description

The routers within the AS are very similar; in fact they are represented with the same source code object. However, gateway routers have an empty pointer to a vector, while the internal nodes populate its vector with a random number of nodes. Each internal router is initialized separately from any other internal router and thus may have any number of nodes associated with it. As with the previous objects, the number of node objects associated with each internal router is governed by a pair of simulation constants.

The scaling factor that was mentioned above in AS Description also applies to the number of nodes. However, in this instance the effect of the scaling is reverse; the AS objects closer to the leaves will likely have more nodes than those AS object closer to the root.

Routers are randomly marked as "reactionary", which indicates that the router is allowed to perform the special set of functions and abilities that are the focus of this research. This marking is performed by randomly selecting offsets into the global array of routers. If the randomly selected router is already marked as reactionary or it does not fall within the acceptable depths of the network, the selected router is not marked. This marking process is continued until enough routers have been marked to satisfy the simulation constant regulating the percentage of routers that are to be granted the special reactionary abilities.

If a router is not marked as reactionary, then it will simply forward all packets that it encounters and drop all packets once its queues are full. Reactionary routers will take different courses of action based on the specifics of this investigation. The number of routers that are marked as reactionary as well as the range of levels within the network

structure that they may exist is determined by network simulation constants (see APPENDIX – CONSTANTS FILE). A reactionary router will refrain from exercising its special abilities until a specific number of malicious packets have been forwarded. After having forwarded the requisite number of packets, the router is then assumed to have "learned" the specific pattern of the threat. The terms signature and pattern are used throughout this research interchangeably despite the difference between signature that exist in data and behavioral patterns. The association of the previous terms is part of the assumption that a perfect system exists in which the presence of a worm can be detected. The research within this paper has set the number of malicious packets to 5. Thus, once a reactionary router has forwarded 5 malicious packets, the router will immediately begin invoking its special countermeasures. These special actions may start in the middle of a time step; the router may exhibit two different behaviors during the same time step. The selected number of malicious packets that must be processed before reacting to the threat (5) may appear to be a small number; however, assumptions used in the creation of the simulation should be revisited. The simulation only creates and processes packets destined for valid nodes. Therefore, routers will react after having seen 5 valid packets, not 5 packets in general. Based on the behavior of the worm that is depicted in this research, the actual number of packets processed by the router would be a great deal more than the 5 required here before any sort of additional actions would be allowed.

2.2.1.3 Node Description

Nodes exist in a list within an internal router and typically represent the vulnerable machines that exist in a network. There is an option that allows for a certain number of nodes to be flagged as invulnerable to infection, but that node does not

contribute to the simulation as it will never create or accept packets that are being simulated. The frequency with which an infested machine attempts to infest a valid machine is another simulation parameter that is discussed in below in Worm Propagation. A pair of simulation constants determines the rate at which a compromised node attempts to spread. This range of values holds constant regardless of what level the node exists at within the network hierarchy.

When a node that was once clean becomes infected, there is a simulation constant that determines how many time-steps the machine remains clean. This additional optional delay is to allow for the modeling of such worms that are not instantaneous when compromising a host.

Nodes are randomly infected after the network is complete. There is a simulation constant that determines the absolute number of nodes that should be infected within the network at time step zero. After the network has been completely built and all relationships between physical entities have been established, nodes are selected at random from an array that contains all the nodes in the simulation. If the selected node is already infected, another selection is made. If the node is not infected, then a counter is incremented and the loop continues until the simulation parameter is satisfied. The selection is not governed by any simulation constants; initially infected nodes may exist at any level within the network.

2.2.1.4 Physical Entity Summary

The physical entities and their relationships were designed to allow for the broadest possible range of applications, models and interpretations. A straight hierarchical approach to network design applies to both corporate intranets [7] as well as

a general description of the modern Internet [16]. Simulation parameters can be tweaked to simulate a small business organization as easily as that of a large network. Much larger applications could also be obtained if one interprets node objects as also representing an infected internal AS that contains at least one infected machine. Such generalizations and adaptability allows for a general simulation of grand proportions.

A sample diagram of how one AS object may be randomly generated is shown below in Figure 2.

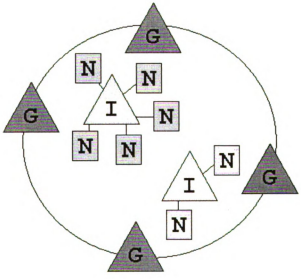

Figure 2: AS Example A sample AS object with gateway routers (G), internal routers (I) and nodes (N).

2.2.2 Logical Entities

The logical entities are similarly designed with simplicity and the ability to abstract in mind. The elements discussed in this section are the worm and the packets that traverse the network simulation (including packet routing).

2.2.2.1 Worm Propagation

After the initialization of the physical entities, nodes are randomly marked as infected. The total number of randomly selected nodes depends on a simulation constant that represents an absolute value, not a percentage.

All node objects consist of a set of variables that act as timers for various events, one of which dictates the rate at which the infected node will attempt to spread to another machine. The timer value is determined by a set of simulation constants and specifies the number of time steps that the node will remain dormant until it attempts to infect another machine within the simulation. Once the timer reaches zero, the node chooses a destination at random. The arbitrary destination is determined by a random number selected between zero and some multiple of the total nodes within the virtual network. If this random number corresponds to a valid node that was created during system generation, then an attempt is made to infect the randomly selected node. After the random destination has been made, it is considered an attempt to further propagate and the internal node timer is reset; failed attempts are not allowed to try again.

If the node successfully selects a peer to infest (the target), the node creates a packet destined for that target and passes the packet to its parent router to have the packet delivered. The details regarding the construction of the packet and how it is routed through the simulation is detailed below in Packet and Routing Description. For example, if the total number of nodes is 10 and the scaling factor is 2, the random number will be between 0 and 19 inclusive. This effectively results in a 50% success rate per infected node.

When a node receives a packet there is another timer that is started that represents the time it takes for the node to be compromised once contact has been made. After this timer expires, the node is marked as infected and its own internal timer is set that determines the rate at which this newly infested node will attempt to spread. At this point the node will behave exactly as described earlier in this section. This cycle repeats ad infinitum until the simulation is halted or completes all time steps.

2.2.2.2 Packet and Routing Description

Packets are container objects that consist of source and destination node offsets and a list of intermediate routers. Packets are created when a node successfully targets a peer. The source node populates the source and destination offsets and places its parent router first in the list of routers. The packet is then passed to the parent AS for further route building.

The parent AS, and all subsequent AS objects, use a simulation constant that determines the number of intermediate routers that will be added to the routing list. The source AS adds random internal routers to the routing list until the simulation route length requirement is satisfied. The AS then adds a single gateway router if required (the destination AS is different than the current AS).

AS objects maintain a routing table that uses the strict hierarchical structure of the network to determine which AS should receive the forwarded packet. When a gateway router encounters a packet that has no more routers in the routing path, it forwards the packet to the next AS. When an AS receives a packet, the packet first has a gateway router added to its list (the entry router). A set of random internal routers is then added, and if the current AS is the destination AS, the parent router is added to the end of the

list; otherwise, an exit gateway router is added. If the parent router is the final destination on the path list then that parent will forward the packet to the appropriate node when it processes the packet. If the final location in the list is a gateway router, the above process of forwarding the packet to different AS objects is repeated until the destination AS receives the packet and it is successfully delivered to the target node.

During the generation of the packet's path, a simulation constant governs how many intermediate routers in the AS will be involved in transporting the packet. This constant determines the percentage of internal routers that will be added to the path. The percentage is based on the total number of internal routers within the AS and the routers are selected at random. In the experiments that follow, the percentages used are 20% and 2% for the extreme and realistic scenarios, respectively. In other words, for the realistic simulations (2% involvement), when an AS that contains 100 routers receives a packet, it will add on the entry gateway then randomly select 2 more nodes to append to the list, then finalize the path by adding either the parent router of the destination node or a different gateway router (exit router). After the route is completed for the current AS, the packet is then queued within the entry router and allowed to pass through the AS. When it reaches the final router on the routing path, it will either be forwarded to another AS where this routing protocol will be repeated or it will arrive at the intended destination and terminate its journey. When the packet reaches the destination AS, the routing algorithm removes the parent router for the destination node if it exists in the route and adds that parent to the end to ensure that the packet's last intermediate hop is correct. There are logical checks throughout the route building process to guarantee that any single router exists at most once within the list.

Figure 3 provides an example of a packets path under the realistic set of parameters. Further, the example assumes that each AS contains 100 internal routers. The realistic parameter set dictates that the percentage of router involvement in the transportation of a packet is set at 2% (see above for explanation). The infected node exists in the source AS and randomly selects a peer to infect; the shaded node in the destination AS. All other nodes (except for the two directly involved in this example) are assumed not to be compromised in order to simplify the example. The packet is created an handed off to its parent internal router which in turns adds a number of other internal routers to the path of the packet. This amount is directly related to the total number of internal routers within this AS and the percentage that the simulation specifies regarding router involvement. In this example since there are 100 internal routers in each AS, an additional 2 routers are selected at random. Finally, the AS selects a random gateway router to complete this portion of the packet's path. An identical algorithm is followed at each intermediate AS and at the destination AS. There is one exception, however, that may occur at the destination AS: the parent of the destination node is selected at random and is included in the involvement path. In this case, the parent router is removed from the path and added to the end of the path. Figure 3 shows the other case where the parent router is not selected because there are 2%+1 internal routers that transfer the packet. This occurs because the parent was not selected during the random involvement selection and is simply added to the end of the existing path. Also note that this exception always occurs at the source AS: the parent is added to the list first and then the percentage of internal routers is added at random. Thus, if the numerical representation of the

percentage of routers to be added to the path is k (k=2 in Figure 3), the source AS will always have k+1 internal routers added to the packet's path.

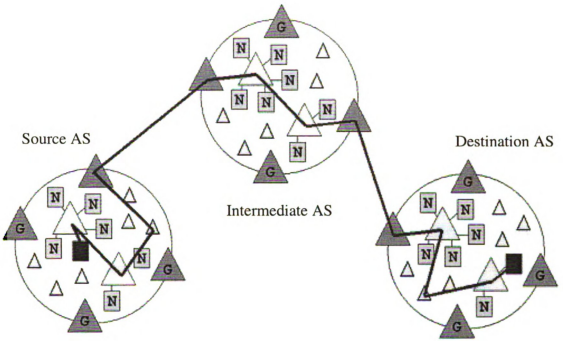

Figure 3: Example packet route Example path taken by a packet originating in the "Source AS", passing through an "Intermediate AS" and infecting a node in the "Destination AS". This example assumes there are 100 internal routers in each AS and uses the realistic simulation parameter of 2% involvement; 2% of the internal routers are involved in transporting each packet.

2.2.2.3 Logical Entity Summary

The simulation concerns itself strictly with malicious packets, failing to account for the normal traffic that one would expect within an actual network. This decision was made for several reasons. First, the focus of the research is directed at how to manage and ultimately stop the spread of a worm and there is no approach presented that would benefit from the inclusion of normal traffic. Second, including the normal traffic would only increase the amount of memory and resources needed to simulate any given

network. Finally, valid traffic, while it experiences surges from time to time is basically consistent from day-to-day [5]. This consistency allows for an assumption of constant valid traffic, which in turn results in router queues only having to model the unused portions of the queue. Analogous to mathematics, this constant valid traffic may be removed without loss of generality. Ultimately, since there is no methodology presented that would benefit from the inclusion of normal traffic and because the simulation assumes a perfect detection rate, there is no foreseen benefit from the adding of benign traffic and it is therefore not considered.

As alluded to in the previous paragraph, there are some assumptions made by the simulation software. First, the simulation assumes a perfect detection rate. That is to say that a router that has been marked as reactionary will *always* detect and perform the necessary reaction. This assumption is made to simplify the problem as detection is a non-trivial issue [14, 17, 1] and is outside the scope of this research. Secondly, the simulation does not simulate any protocols; infestations are achieved via a single packet. Assuming a single packet to allow a node to succumb to the threat is a valid assumption for some worms such as slammer or sapphire [18], but may not suffice for other worms such as Code Red [9]. Despite the direct relationship between packets and infestation, one could also extend the existing abstraction and have the single packet within the simulation represent the final packet in a series required to deliver the entire payload. Third, the network does not drop packets unless the next router in the packet's route has a full queue. This assumption is not accurate for real-word networks; however, increasing the success of packet transmissions generates a worst-case scenario in regards to worm propagation. This serves only to strengthen the findings presented here – if the methods

examined here can combat the spread in a perfect transmission setting, it can do no worse in a real-life situation.

The final piece of the logical system that needs mentioning is the routing scheme. The decision to place the routing algorithm at the AS level was based on many factors. First and foremost, having the AS determine the path lets one easily simulate the asymmetry of the Internet. Second, there exists a wide variety of routing protocols and schemes (RIP, IGRP, RON, OSPF, MPLS, etc) for both inter- and intra-network communications [7]. Selecting one intra-network protocol is difficult as this decision is left to the appropriate administrator. Further, limiting the simulation to one specific algorithm would not allow the simulation to alter the level of network exposure to traffic. In reality, routes change as nodes go up and down and alternate paths must be created. However, despite the small constant changes in individual nodes, the general path remains relatively unchanged as a result of the basic hierarchical structure of most networks and ISP policy that prohibits handling traffic that is not their responsibility [7]. As a result, the author feels that the current random path is an adequate and appropriate means to properly simulate network traffic while not specifically attempting to model one particular protocol.

The simulated logical components of the network are not completely true to their physical counter parts, but this is not the intention of the software. The software is to model as closely as possible the network and the interactions of the propagation of the worm and the various routers that it encounters on its journey. Therefore, the simulation need be exactly that - a simulation, not emulation.

CHAPTER 3 - EXPERIMENTATION

All experimentation was conducted on the authors home PC, a 2.8 GHz Pentium IV Win XP Pro with 512 MB RAM. The software was written and compiled under Microsoft Visual Studio .NET 2003 (version 7.1.3088) in C++. Repeated runs of the application were achieved through a Perl script running under Cygwin [3].

Unless otherwise stated, all results presented are the averages of 1000 runs of the simulation performed with the exact same simulation constants but with different seed values. Specifically, the seed values are the numbers 1 through 1000, inclusive. The standard deviation for the extreme simulations regarding network infection percentage averaged roughly 3.5%. Standard deviations for network infection percentages ran under the extreme scenario averaged 3.2%. There is also set of milestones that are recorded for each averaged set of results that provide a set of criteria on which each set of experiments can be measured.

Further, unless otherwise stated, all experiments were conducted with as many identical simulation constants as possible; only factors governing the modified behavior and total simulation time were altered between sets of experiments. The maximum number of AS and router entities were 256 and 1024 respectively. The actual number of nodes created in the simulation was about 9,000 on average. The k-ary tree of AS objects was limited to a depth of 5 and actual simulations exhibited AS objects reaching maximum depths of 4 or 5 [16]. These maximum depths correspond to the various layers discussed in [16]: dense core, transit core, outer core, regional ISPs and customers. The parameters to create the tree directed each AS to randomly generate child AS objects over the range [8,12] with numbers of routers being reduced as depth in the tree increases and

vice versa regarding nodes [19]. Infected nodes attempted to propagate every 5th time step and routers required 2 time steps to process a single packet. Simulations were ran for a maximum of 1,000,000 time steps and were shortened when possible to increase the granularity of the recorded statistics.

3.1 Base Case

The model developed for the purpose of this research was used to simulate a completely unprotected network of computers to establish a base case. This base case, along with common milestones, will be used to help quantitatively measure the effects of the various techniques investigated later in the paper. The base case also serves as a basic, unmodified network of components that accurately portrays the majority of the current infrastructure of the Internet. Therefore, the results of the base case can be compared against actual measurements regarding the spread of past worm epidemics in addition to theoretical models of worm behavior. Figure 4 presents the theoretical results using the biological epidemic model discussed in [2] and Figure 5 shows the statistics resulting from simulations of a Code Red like worm [9]. Lastly, Figure 6 is the average result obtained from 1000 simulations using the simulation software created for this research under both the realistic and extreme parameter settings. There is a slight difference in the when the behavior of the infection is exhibited because the extreme case utilizes more routers in each path and thus it takes the worm slightly more time to traverse all the intermediate hops on its path.

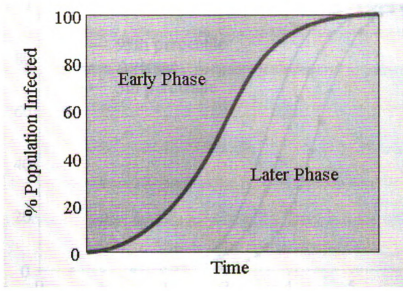

Figure 4: Theoretical Results from [2] The spread of a worm as predicted by the biological model discussed by Chen, et al.

Comparison between the theoretical and previously simulated results implies that there are slight differences in the transition from the "early" to "later" phases. The theoretical model results in a smoother transition between phases where the simulated measurements reveal a sharper increase in infestation once the epidemic gains momentum. Observing the average results measured from NetSim shows that the simulation also shows a rather sharp increase in infestation once the worm gains momentum, more closely resembling the results obtained in previous simulations. Thus, Netsim is considered to successfully model the propagation of malicious code because it exhibits the same general behavior observed in both models. Additionally, NetSim arguably models the real world simulation more closely then the theoretical model and may be a better tool than the mathematical approach.

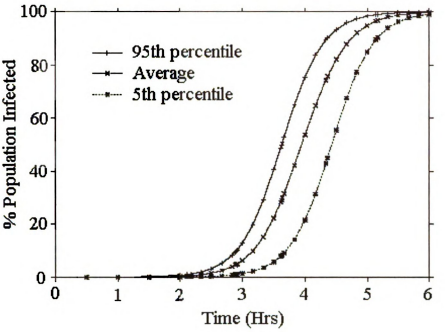

Figure 5: Results from Code Red like simulations from [9] The simulations performed by Moore et. al. modeled the behavior similar to Code Red. The three lines represent the 5th, average and 95th percentile of 100 simulations.

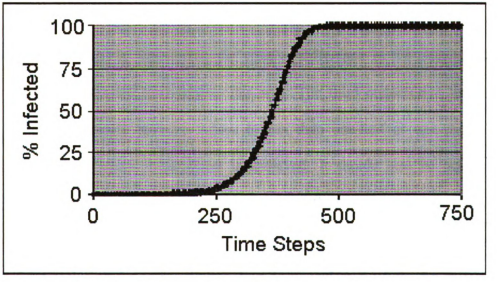

Figure 6: Base case results from NetSim The results from running NetSim under the extreme and realistic parameters settings.

The milestones that are applied to the results of each simulation are presented below for the base case in Table 1. The complete simulation is divided and statistics are recorded every X time steps and is denoted in the table by the number in parenthesis after the short description on the experiment. At each time step designated as one that should result in a statistical measurement, a set of values are recorded to a flat file regarding various aspects of the current state of the simulation. These aspects include the number of nodes that have been infected, the number of routers that have become aware of the invasion, and the total number of worm packets in the system to name a few. Homeostasis (indicated by 'H' in the table below) is computed by converting those raw numbers into percentages. For example, if there are 1000 routers total in the simulation, but there are only 500 routers that have been granted the ability to react to the threat, then all the statistics regarding the number of routers that have become aware of the worm are taken with respect to 500. That denominator is used because 500 represents the total number of routers that could possibly become aware of the problem The value for homeostasis is determined by calculating the relative difference in the corresponding percentage from one recorded time step to the next. The third consecutive time step where the relative difference between values is less than one-half of one percent is considered the point at which homeostasis is achieved. It should also be noted that limitations in the graphing software placed a limit of 250 time steps on which values could be recorded. This is important because as the various methods reduce the spread of the malicious code, the total running time is increased, and as a result, the difference between recorded time steps increases proportionately. Finally, the percentages (33%, 67% and 99%) contain the recorded time steps in which the simulation first either met or

exceeded that percentage. These values will be basis upon which all future experiments will be measured against. Later data sets will present the raw data as well as the relative value to those observed for the base case.

Table 1: Base case milestones This table shows the time steps at which the milestones were first observed to meet or surpass. The number in parenthesis in the column header indicates the intervals (in time steps) that data was collected.

Milestones		base-case (4)
Infected Nodes	*H*	456
	33%	344
	67%	388
	99%	460

3.2 Traffic Throttling

The first set of experiments focus on the effects of slowing down the malicious packets as they traverse the network. The slowdown is accomplished by altering the length of time that the router requires to process a packet. This alteration may only exist within a router that has randomly been designated as a reacting router. When routers are first created during the random creation of the network, the router is given a base processing delay according to the following rule ('this' refers to a router object):

```
this->base_proc_delay =
      ROUTER__PROC_DELAY *
      (this->depth + 1);
```

Immediately after this line of code, a member variable denoting the actual processing delay is assigned the value of the base processing delay. This 'actual' processing delay is copied to all incoming packets. Therefore, there is an element of scaling based on the depth of the router within the network. The routers near the top of the hierarchy do not need as much time to process packets as do those routers nearer the leaves. When a reacting router encounters the prerequisite number of malicious packets such that it "learns" the signature of this particular intruder, the actual processing delay is altered based on the simulation constant as shown below:

```
this->actual_proc_delay =
      this->base_proc_delay *
      ROUTER_THROTTLE__DELAY_FACTOR;
```

In the experiments conducted for this research, the value of ROUTER_THROTTLE__DELAY_FACTOR was set at 2 and 20, thus routers would hold on to malicious packets for twice as long as their peers after having "learned" the signature of the worm when set to 2.

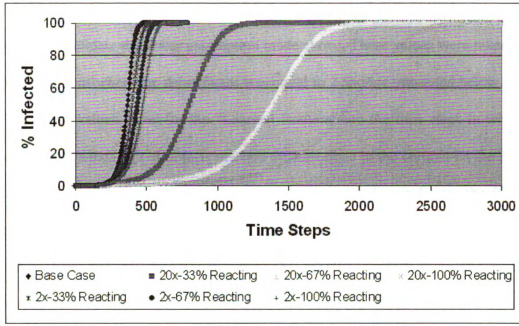

Figure 7: Throttling defenses with base case The graph shows the two levels of throttling along with the base case. All were simulated under the extreme set of simulation parameters. The legend first describes the level of throttling (either 2 or 20) then indicates how many of the routers were allowed to react to the threat.

The throttling scenario was repeated three times varying the percentage of routers that were randomly marked as reactionary and those results graphed against the results observed from the base case, see Figure 7. The graph clearly shows that the worm achieves complete infestation regardless of how many reactors exist in the network. Despite the inability to prevent the complete domination, throttling does delay the apparent inevitable. There is a direct correlation between th number of reacting routers and the degree to which the worm progresses throughout the network. The correlation is directly related to the number of reacting routers within the simulation and results in a shifting the time for infestation in the positive X direction while not impacting the ultimate degree of infestation at all. Throttling will give additional time for some other means of countermeasures, but will not provide any automatic protection.

Additional measurements identical to those recorded for the base case were performed to validate the linear assumption made above. These values were taken for the set of simulations where the delay factor was equal to 2 and are presented below in Table 2. The integer values listed for the "Learned Routers" are the time steps in which that corresponding percentage of objects reached the milestone in the column on the left hand side. The integer within the parenthesis in the column heading indicates the frequency with which statistics about the simulation were recorded. The percentages in the body of the table show the difference in reaching the identical milestone in the base case simulation. Relative change is computed with respect to infected nodes for the reactionary routers because no such routers existed in the base case.

The values observed for the throttling experiments strengthen ones beliefs based on examination of the graph. The milestones are consistently reached with about 10% additional time when 33% of the routers react, approximately 20% additional time for 67% reactionary routers and roughly 30% more time when all routers in the network are reactors. Similar consistent shifts were exhibited from the spread of router learning in comparison to the spread of infected nodes.

It is worth noting here that the percentage increase in the spread of knowledge versus the spread of the worm holds at approximately 20%; the same percentage specified in the simulation constants that determine the number of internal routers that constitute any path that must travel through an AS. The author makes no claims regarding this possible correlation and recommends further testing to isolate this parameter and its effect on worm progression. Such experimentation is outside the scope of this research and is left to the initiated.

As interesting as the results may be – this technique fails to stop the worm from achieving reaching complete infestation of the network. The result is not unexpected because the reacting routers do not actively remove malicious code from the network. Therefore, all intended targets can and will be reached if given enough extra time. Unfortunately, the time afforded by this technique does not provide enough benefit to protect any portion of the network. Using time scales in the most extreme cases, worms spread through out the Internet in as a little as 20 minutes [13, 14]. Thus, a delay of 32% equates to approximately 7 additional minutes to respond – not enough time to allow for administrators to deploy (let alone develop) a patch for basic routers to stop the spread of the worm.

The result does demonstrate that knowledge of the outbreak can spread throughout the sub-network of reacting routers more quickly than the infestation of nodes. The increased speed in the dissemination of knowledge is a direct result from how a reacting router may "learn" a signature and it is worth repeating here. A reacting router learns from not only the packets that it delivers and initially accepts from its children but also from the traffic it processes. This creates a situation where a packet that interacts with two nodes now interacts with many nodes along the packet's path. The experiment shows that taking advantage of this paradigm proves useful in that many routers are repeatedly encountered as the worm spreads throughout the network. Thus, if reactionary routers could eliminate the suspect traffic rather than just slowing its progress, then perhaps what was once considered the inevitable may become preventable. A useful approach must permit network devices to remove traffic on behalf of the network.

Table 2: Throttling Milestones The values compare the times at which milestones were achieved in the various simulations with respect to when those same milestones were achieved in the base case simulation. The percentages indicate the percent change from the base case.

Throttle		base-case (4)	throttle-33 (4)		throttle-67 (4)		throttle-100 (4)	
Infected Nodes	*H*	456	508	11.40 %	556	21.93 %	600	31.58 %
	33%	344	376	9.30 %	412	19.77 %	444	29.07 %
	67%	388	424	9.28 %	468	20.62 %	508	30.93 %
	99%	460	508	10.43 %	560	21.74 %	608	32.17 %
Learned Routers	*H*	456	360	-21.05 %	380	-16.67 %	396	-13.16 %
	33%	344	256	-25.58 %	264	-23.26 %	276	-19.77 %
	67%	388	292	-24.74 %	308	-20.62 %	324	-16.49 %
	99%	460	356	-22.61 %	376	-18.26 %	396	-13.91 %

3.3 Content Filtering (Packet Dropping)

Observing that the malicious packets are not actually removed from the active packets within the network leads one to develop a means that eliminates these packets. The first method that is examined in this research was a basic form of content filtering. In this approach, once a router has learned the signature of the worm (as before this occurs after having forwarded five malicious packets) it simply discards the packet. The packet that got forwarded to the next appropriate hop is completely removed from the network and all consideration. The router has no other special abilities and does not make any attempt to communicate its acquired knowledge with any other entities in the network.

The milestone data collected from the experiments (Table 3) shows an obvious result from removing the packets from the network: certain levels of infestation are never reached. The average number of nodes infected only exceeds 33% with only one-third of the routers reacting. The length of the time required to obtain homeostasis also greatly

increased to approximately 10 and 100 times the base case for one-third and two-third reacting routers, respectfully. This linear relationship relating reacting routers to time required to achieve homeostasis does not maintain through out the experiments – a complete network of reactors converges in nearly half the time than that of the base case.

One more interesting observation from the comparative data is the level of involvement amongst the reacting routers. The relative percentage of routers that learned the signature of the worm during the outbreak consistently decreased as the number of reacting routers in the network increased. On average, with one-third of the reactors available as reactors, less than 98% of those entities become involved enough such that they learn the signature. This level of involvement lessens such that it reaches only 33% with two-thirds of the routers reacting and does not reach any milestones at all when the network is completely comprised of reacting routers.

Further examination of the current experiment was conducted to get a more accurate account of both the infestation as well as the required involvement of the reacting routers. To achieve this additional insight, the maximum values of infected and learned router counts were collected and are presented in Table 4, below. The values in parenthesis represent the actual percentage of routers within the network that had learned the signature, as opposed to the percentage outside the parenthesis which is a value taken with respect to the reacting router population. Focusing on this particular fact, it can be seen that despite the fact that the two-thirds reactors approach never reached 67% involvement, it actually utilized approximately the same number of routers with respect to the entire network. Additionally, the latter approach limited the infection from the worm to less than 20% of the total number of nodes.

Table 3: Drop Milestones These values are determined and represented in exactly the same manner as those values in Table 2. The single hyphen indicates that the milestone was not reached and therefore no value was obtained and thus no percentage to compute.

Drop		base-case (4)	drop-33 (27)		drop-67 (445)		drop-100 (4)	
Infected Nodes	*H*	456	4725	936 %	45835	9,952 %	236	-48 %
	33%	344	2655	672 %	-	n/a	-	n/a
	67%	388	-	n/a	-	n/a	-	n/a
	99%	460	-	n/a	-	n/a	-	n/a
Learned Routers	*H*	456	1980	334.21 %	32930	7,121 %	280	-39 %
	33%	344	450	30.81 %	16020	4,557 %	-	n/a
	67%	388	900	131.96 %	-	n/a	-	n/a
	99%	460	-	n/a	-	n/a	-	n/a

Table 4: Drop Maximum Values The maximum values observed relating to the percentage of nodes in the network that were infected as well as the percentage of routers that had became aware of the worm under the extreme parameter set. The first percentage is with respect to the subset of reactionary routers; the value in parenthesis is with respect to all routers in the entire network.

Maximums	Infected Nodes	Learned Routers
drop-33	66.0%	98.6% (32.5%)
drop-67	19.5%	57.6% (38.6%)
drop-100	0.2%	2.4% (2.4%)

The saturated simulation produces results that are remarkable in comparison to the others. The 33% and 67% reacting router simulations create an environment such that paths exist of entirely normal routers between an infected node and one that has yet to be infected. These paths that have no reacting routers constitute a hole in the security of the network. When there exists a sufficient number of compromised nodes in the network, these vulnerable holes are exploited, resulting in the further spread of the worm.

The aforementioned theory regarding the exploitation of vulnerable paths can be seen in Figure 8. The curve that is expected as a result of the infestation (as demonstrated by the base case) is flattened greatly, nearly to a point that it is unrecognizable. The percentage of routers that learn the signature increases dramatically. However, the worm continues to spread further throughout the network after the set of reacting routers nearly completely becomes aware of the problem. The outbreak actually exhibits its steepest increase in penetration while the number of learned routers is near its highest point. The maximum infestation percent presented in Table 4 closely resembles the number of nodes within the network that are located directly under a reactionary router. The percentages of nodes that are direct children of reactionary routers when one-third of the router population is altered to be a reacting router is 32.9%; when there are two-thirds of the routers reacting, the percentage of nodes that are direct children rises to 66.99%. Thus, the protection appears to be limited to those nodes that are directly connected to routers capable of offering protection from the threat. The configuration where two-thirds of the routers are reactionary provides additional protection to those nodes that are not directly under reactionary routers because less than 20% of the nodes in the network are infected at the end of the simulation yet over 30% of the nodes were not directly under a reactionary router.

The previous result in conjunction with the data in Table 3 and Table 4 implies that it is not the actual number of routers that may react to malicious traffic as much as it is the location with respect to the source(s) of the outbreak. The simulation that allowed two-thirds of the network routers to react provided a wider range of points on which the network could fight back. The network had fewer "holes" and as a result fewer nodes

were initially infected. This reduction in numbers reduced the power of the brute force behind the attack, and the level of infestation converges are slightly under 20% of possible nodes.

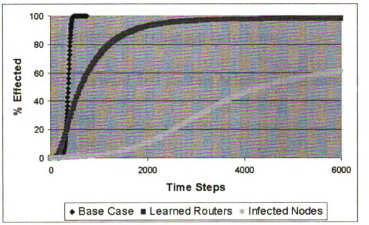

Figure 8: 33% Reacting Routers Filtering Content The graph is obtained under the extreme parameter set and shows that routers become aware of the invasion very quickly. However, even after nearly all possible routers are reacting, the worm continues to spread throughout the network.

The saturated simulation effectively stops the outbreak, but it is surprising how little of the network was utilized to achieve such an effect. With only about 2.5% of all the routers in the network acquiring the signature information of the worm, the incident was contained to one-fifth of one percent. This scenario provides a best-case analysis where by all machines can react and demonstrates that a complete solution could be accomplished with as little as less than 3% of routers involved in the counter-measures.

The results from the experiment seem exceptional; that such a remarkably high percent of the network was effectively quarantined as a result of reacting routers. This incredibly successful performance is a direct result of the assumptions made regarding

the hypothetical defense system. The defense system assumes that network traffic contacts (or at least shares information about its payload) with 20% of each AS it encounters and that routers may positively identify (without exception) threats by being exposed to 5 malicious packets. These two assumptions are not typical of a modern system [7] and to simulate a situation that more closely resembles today's practices and technology, simulation parameters were changed as follows:

- Routers must forward 500 packets before it is assumed that the signature has been learned
- The percentage of routers in an AS that are involved in transporting packets is reduced from 20% to only 2%.

The adjustment made to the percentage of the AS that is involved in the transportation of packets is more indicative of today's networks where speed and throughput is typically the focus [7]. Worm recognition, however is still an open question and placing any sort of bounds is technically inaccurate, yet 500 packets is a much more realistic estimate of technology that may exist in the near future than the previous value of 5.

Figure 9 shows the results of the model in which the parameters were more realistic along with a new base case under the same realistic parameters. The worm manages to invade nearly the entire network; achieving a final infestation of 97%. The worm exhibits a rate of infestation that is close to the base case with slight abnormalities in the ending stages as a result of enough routers learning so that they may make an impact on the worm's proliferation.

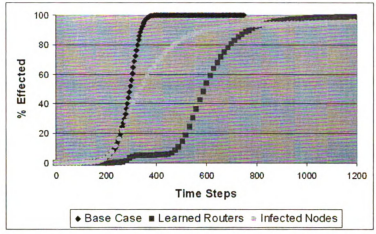

Figure 9: 33% Reacting routers with a more realistic set of parameters
The network is not sufficiently defended as the worm swells to nearly 100%. The reactors first respond to the initial source, but it is too late to avoid a massive outbreak.

Figure 10 shows all six infection curves for both simulation parameter sets and reactionary participation in addition to the realistic base case. The realistic simulations all conclude within the time frame but 33% and 67% extreme simulation results extend beyond the limits of the graph. The 100% extreme case barely exhibits any infection at all and thus converges very early on. The graph clearly shows how all three realistic simulations diverge from the base case curve at the same point (divergence point) and have varying degrees of growth dependent upon the amount of routers that are actively trying to stop the infection. The more routers there are dropping malicious packets, the flatter the curve after the divergence point.

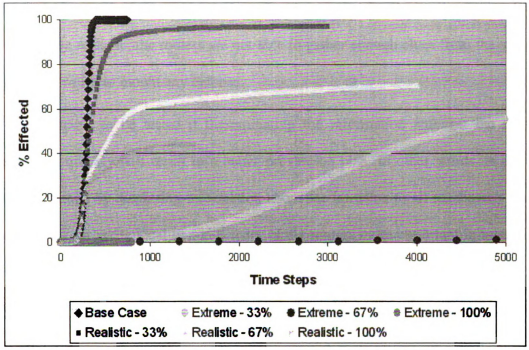

Figure 10: Collection of Infection Curves for Drop Simulations The graph combines the realistic base case curve with the six simulations regarding basic dropping: the two simulation parameters and the three levels of router participation. The realistic simulations follow the base case to a point, the divergence point, and then they start to exhibit the influence of the defense; more routers fighting back result in a flatter infection curve after that point.

In the early stages of the outbreak, shortly after the worm begins its rapid ascent into the network, the number of routers that are aware of the worm's presence jumps. The increase, however, is short-lived and levels off for a period of time. The increase occurs near the source of infection as a result of the node that is continuously emitting packets into that region of the network. That initial source is sending its malicious code to all corners of the network and successfully disperses itself so that the other regions are not exposed to enough traffic to be able to recognize and prevent the threat from continuing. In the previous simulations, the accuracy and efficiency with which the routers could successfully identify and eliminate the malicious traffic condensed the routers response

curve so far as to eliminate this behavior. Conversely, under the parameter set that is currently more realistic, the routers are not able to gather enough clues from the sporadic nature of the worm to mount any defense.

It remains to be tested if disseminating the information amongst the reacting routers more quickly will result in a better prevention plan. In other words, is there an advantage gained by dispersing information about the intrusion to all reactionary routers more quickly? Or is the locality and density of the routers more important in mounting a defense?

3.3.1 Fire with Fire

From the previous experiment, it is obvious that eliminating the malicious code provides an obvious benefit. The specific numbers of reactionary routers throughout a given simulation remains constant, thus the focus of the experimentation becomes to facilitate the knowledge of the breach as quickly as possible to all appropriate routers. The previous simulations utilized a defensive theory whereby the router only learned of the threat once it had encountered it several times (5 for the purpose of this research, but this value is a user defined simulation constant). This simulation introduces inter-router communication henceforth referred to as "signal" or "signaling". This additional communication provides the infrastructure for a revolutionary approach to combating the spread of a worm; fire with fire.

As the name implies, the router acts in a manner similar to an infected node. Once a router learns the signature of the worm, it sends out a signal to a random router within the network. This learned router continues to send signals to random destinations for the duration of the simulation. When a reactionary router that has not yet learned the

signature receives a signal, the router instantly acquires knowledge of the threat and can defend against it. Further, the delays of the signaling packets are proportional to the delay of the worm packets at that particular router. These delays are computed when the signaling packet is received based on the current delay for the worm traffic. This proportional value is determined by a simulation constant and for the purpose of this research is held constant at 1.

The milestone information is presented below in Table 5. The data shows relatively little change in the maximum progression of the worm as compared to the previous milestones obtained when a purely defensive packet filtering technique was used. The one-third reactionary router allotment is the only configuration where the percentage of infected nodes exceeds 33%. An additional point of interest to take from the data is that homeostasis of the infected nodes occurs at relatively the same time despite including the ability of routers to communicate amongst each other. Unfortunately, the increase in network traffic and inter-communication does not provide any real benefits with respect to the percentage of nodes fallen to the worm invasion. Actual maximum values and more precise measurements concerning infestation and convergence are discussed later.

While the progress of the worm was not greatly altered by the addition of inter-router communication, the dissemination of knowledge between routers flourished. In comparison to the previous experiment where routers stop learning once the invasion was contained, this approach continues to share knowledge and quickly reaches 100% router awareness very quickly. The table does a poor job of distinguishing milestones between

time steps because the routers acquired the knowledge so quickly with regard to the spread of the worm that the intervals were too large to show any distinction.

Table 5: Fire with Fire Milestones

Drop-fwf		base-case (4)	drop-fwf-33 (45)		drop-fwf-67 (445)		drop-fwf-100 (4)	
Infected Nodes	*H*	456	5264	1,054 %	61410	13,367 %	180	-60.53 %
	33%	344	3195	828.78 %	-	n/a	-	n/a
	67%	388	-	n/a	-	n/a	-	n/a
	99%	460	-	n/a	-	n/a	-	n/a
Learned Routers	*H*	456	450	-1.32 %	1780	290.35 %	236	-48.25 %
	33%	344	225	-34.59 %	445	29.36 %	164	-52.33 %
	67%	388	225	-42.01 %	445	14.69 %	184	-52.58 %
	99%	460	270	-41.30 %	445	-3.26 %	224	-51.30 %

The results are impressive despite the lack of granularity. The router network successfully propagates its knowledge about the intruder much more quickly than the intruder itself spreads throughout the network. The experiments were run a second time over a shorter time period to explicitly examine the rate at which the routers learn about the intruder. The results of this additional experimentation are shown below in Figure 11. The x-axis corresponds to time steps while the y-axis represents the percentage of reacting routers that have learned the worm's signature. The base is included as well on the graph for reference. The increase in the flow of information amongst routers as opposed to that of nodes is obvious. The routers learn of the signature in approximately half the time. Adding more routers as reactors does not add to the amount of time required for the routers to become aware of the outbreak, in fact, the more reactors there are, the less time it takes for all of them to learn. Further, it appears that there is not a direct relationship between the percentage of routers and the rate at which they all learn;

the difference between one-third and two-thirds reactors is much larger than the difference between two-thirds and complete participation amongst the routers.

The results presented in this section are important for at least two reasons: 1) this demonstrates that it is possible to disseminate information about a worm faster than the worm can penetrate the network (regardless of the percentage of routers participating as reactors) and 2) despite having nearly all possible reacting routers aware of the signature, the worm can still successfully exploit the small existing holes in network paths and continue to propagate.

The speed at which this paradigm disseminates knowledge about the worm outbreak is impressive. This approach successfully spreads knowledge at all participation levels much faster than the worm itself spread in the base case. Further, the experimentation shows that increasing the number of routers that may react to sensing the threat allows the information to be shared more quickly. This is a result of having a proportionately higher number of sources throughout the simulation as well as a higher probability that the packet will encounter a reacting router. That coupled with the mathematical fact that all routers have the ability to react, and the network's defense mechanism achieves remarkably well. It is conceded that such a situation is purely hypothetical as complete cooperation and universal standards within the hardware community and service providers is far from a reality.

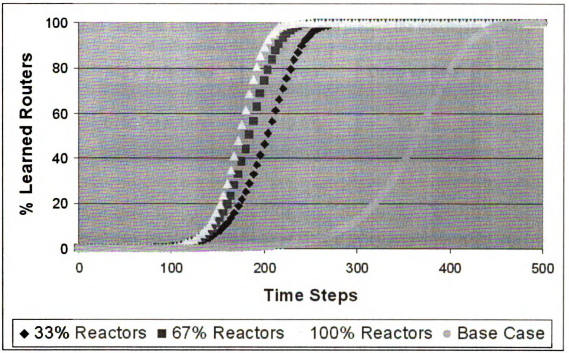

Figure 11: Reactor Knowledge Propagation The knowledge about the infestation spreads much faster than the actual infestation. The negative shift in the X component here is considered an improvement because we are measuring the percentage of routers that have recognized the worm. Reacting routers learn of the threat faster than the worm could have spread without any obstacles or defense.

The continuing spread of the worm after the sub-network of reactors had basically achieved its peak might be a concern. The worm successfully spread and achieved similar levels of infestation to that of the previous experiment where the routers were purely defensive in nature. The maximum levels of infestation achieved for this simulation as well as the previous experiment are shown below in Table 6.

Table 6: Drop & Fire with Fire Maximum Infestation These values were obtained using the extreme parameter set and present the maximum levels of infestation achieved during the simulation

Max Infested	Drop	Drop w/ FwF
33% Reactors	66.0%	65.0%
67% Reactors	19.5%	13.0%
100% Reactors	0.1570%	0.1018%

The results are bit concerning because there is no noticeable benefit from creating routers that add a significant amount of traffic to the network. The reacting routers must consume time and resources locally to properly execute whichever algorithm is implemented to correctly identify malicious code. It must then consume network resources and further local resources when it crafts a signaling packet destined for some other location that may or may not benefit from this additional work (the signal may or may not even be destined for a reacting router). The experiment does not provide conclusive evidence of any kind that leads one to believe that the additional work was a sufficient trade off between resource consumption and worm prevention. The worm successfully discovers and abuses the paths that exist between nodes that do not contain reacting routers that can proactively defend against its proliferation.

There may be a slight improvement in defense when the reactionary routers mimic the behavior of an infected node. However, it is the opinion of the author that the slight increase in preventing the circulation of the malicious code is not worth the additional burdens placed on the network. Furthermore, the routers are without any intelligence and theoretically could be sending signals for various types of worms ad

infinitum. A model that addresses the concerns about network utilization and bases its response off the presence of the worm is described and simulation below in section 3.3.2.

This experiment was repeated for more realistic parameters just as the basic content filtering scheme was repeated in Section 3.3. The realistic parameters pertaining to the router reaction time and path length within an AS are identical to the values used in the previously: 500 malicious packets and 2%, respectively. Additionally, there have been adjustments made to the frequency with which the router can fight back. In the extreme case examined above the router would signal on every time step while the worm was restricted to signaling at most every 5 steps. Such a policy in actuality would result in large amounts of traffic and additional network strain, not to mention a router would be occupied with sending out signaling packets and not efficiently route normal traffic. There would be an unacceptable reduction in network usage. This round of simulations reduces the burden on the router and introduces another simulation constant that determines the frequency that the router will send signals to other randomly selected routers. This experiment sets both the rate at which the router and the rate at which an infected node will spread at 5 time steps.

Figure 12 shows the more realistic fire-with-fire simulation along with the more the realistic base case for having one-third of the routers participate in the defense. It can be seen that the initial stages of the worm's infestation closely mimics that of the base case as the number of routers that have learned of the invader is practically zero. However, as the worm gains portions of the network it also increases the amount of traffic that allows reacting routers to learn of the activity. This additional data need only result in a few routers to become aware because each router behaves much like an

infected node; only the routers disseminate countermeasures. Figure 12 shows the point at which the worm deviates from the base case coincides with the rapid spread of router awareness. The routers quickly share the information about the worm effectively quarantine about 21% of the network. This is a slight decrease from the levels observed in Table 6, which is to be expected because information is not dispersed throughout the network as much during packets transmittal and the routers require 100-fold more time to recognize the malicious packets.

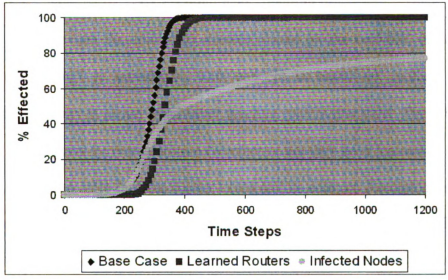

Figure 12: Plausible Fire With Fire simulation with 33% reacting routers Reacting routers quickly respond to the worm and significantly altersthe worm's progress

The more realistic model exhibits the benefits of sharing information between reactionary routers. In the extreme case examined earlier in Figure 11 and Figure 8 the information amongst the routers spread much more quickly than the worm, and as a result, the routers had quarantined just about all that they could before the worm started its aggressive progression into the network. The more realistic models, on the other hand

(Figure 9 and Figure 12), show how communication between routers can alleviate the delay in the reactionary curve (Figure 9; time 300 through 450). While this approach does successfully increase the preventative abilities of the reacting routers, it does so at the possible expense of network resources and has no means with which to stop signaling their peers. A model that allows reactionary routers to successfully signal its peers and has a mechanism to control the number of signals being generated is needed. A possible defense schema that attempts to resolve those issues is discussed and simulated in section 3.3.2, below.

3.3.2 Intelligent Signaling

To help alleviate the additional stresses placed on the network, the concept of "intelligent signaling" is introduced. This concept is based on another non-trivial assumption: the router is aware of all other reacting routers that exist. The other routers know not of their peers current condition (whether or not they have learned of the current signature), they know only that the reactor exists and its address. Assuming that a single machine would keep track of all other reacting routers through the world is highly unlikely. However, there do exist protocols and overlay networks that allow for many machines to maintain communications and for peers to drop and come back without degrading the performance of the overlay network. These networks are commonly referred to as peer-to-peer (P2P) networks. Examples and further information regarding such networks can be found in [15, 6]. P2P networks are highly scalable and easily accommodate members who drop out and rejoin the group. While this is not necessarily a constant issue (that routers go up and down), it is definitely an event that cannot be

prevented. The overlay network should be capable of adapting and functioning in such an environment.

This experiment utilizes intelligent signaling and attempts to limit the traffic on the network by only sending a signaling packet when the router processes a packet of malicious code. This technique has the advantage that it scales directly with the spread of the worm. As the worm acquires more and more nodes and thus creates more and more traffic, the reacting routers respond in turn. Each reactor checks each packet and if it has learned of the signature, it will send out a signal to a random router *and* drop the malicious packet. The solution scales precisely with the worm and prevents routers from unnecessarily flooding the network with signaling packets.

As before, the signaling packets instantly change the receiving reacting router into a router that has learned of the current signature and will drop malicious packets the next time it processes its traffic queue. The knowledge of the signature will be retained throughout the simulation.

The milestones collected for this set of experiments are presented below in Table 7. The number of time steps to reach corresponding milestones relating to the spread of the malicious code is relatively equal compared to those values obtained from the 'Fire with Fire" simulations. Both scenarios had the number of infested nodes converge in about 1000% more time than the base case with reacting routers accounting for about one-third of all routers.

Table 7: Intelligent Signaling Milestones

Drop-sig		base-case (4)	drop-sig-33 (45)		drop-sig-67 (445)		drop-sig-100 (4)	
Infected Nodes	*H*	456	5100	1,018 %	53400	11,611 %	236	-48.25 %
	33%	344	3015	776.45 %	-	n/a	-	n/a
	67%	388	-	n/a	-	n/a	-	n/a
	99%	460	-	n/a	-	n/a	-	n/a
Learned Routers	*H*	456	675	48.03 %	3560	680.70 %	720	57.89 %
	33%	344	270	-21.51 %	445	29.36 %	452	31.40 %
	67%	388	315	-18.81 %	890	129.38 %	-	n/a
	99%	460	630	36.96 %	1780	286.96 %	-	n/a

The length of time required for the routers to reach their respective milestones has grown considerably. For instance, the situation where one-third of all routers are reactionary achieves 99% learned saturation at time 270 when fighting "fire with fire"; however, 99% saturation is not obtained until time step 630 with the intelligent signaling. This is an increase of 233%. Further, when the network is constructed of entirely of reacting routers, intelligent signaling does not reach the 66% learned milestone within the maximum number of time steps (100,000). The only milestone achieved for this configuration is 33%, which occurs at time step 452 – an increase of 276% from the corresponding milestone results from the "fire with fire" simulations.

The increase in time required to distribute knowledge about the worm delays the sub-network of reactionary routers from completely becoming aware of the attack – but does this directly effect the quality of the response offered by this sub-network? Table 8 shows the maximum levels of infestation achieved on average for the three configurations of reacting routers and the two primary strategies for disseminating information. The degree to which the malicious code successfully penetrates into the network is virtually identical in either situation

The protection offered to the network is not noticeably improved because unprotected paths between nodes still exist. These paths ere first observed and hypothesized in Figure 8. However, these results do answer the question posed previously regarding the faster dissemination of information and its impact on the degree to which the worm spreads. Under the scales presented here, there is no sizable advantage gained or lost as a result of router communication.

Table 8: Maximum Infestations for extreme scenario These values represent the maximum infestation percentages obtained under the extreme scenario. Refer to Table 9 for results pertaining to the realistic scenario.

Max Infested	Drop	Drop w/ FwF	Drop w/ Sig
33% Reactors	66.0%	65.0%	65.3%
67% Reactors	19.5%	13.0%	13.8%
100% Reactors	0.1570%	0.1018%	0.1467%

As a result of the "Fire with Fire" strategy, learned routers continually send signals regardless of whether or not a malicious packet is processed. This creates a burden on network resources as the routers become just as responsible for a decrease in network throughput as do the worms. The intelligent signaling does not place additional network resources in peril. When a learned router processes a packet that is considered malicious code, the packet is dropped and a signal is generated. The net change in network usage is zero.

Again, the simulation was repeated under more realistic simulations parameters. The parameters were modified similarly to how they were altered for the realistic simulations conducted in section 3.2.1; the network utilizes only 2% of the routers in an

AS to transport packets and reacting routers require 500 malicious packets to be forwarded before learning the about the worm. The result of having one-third of the routers reacting to the worm under the plausible simulation settings is shown below in Figure 13.

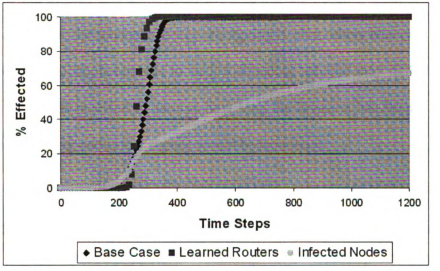

Figure 13: Realistic Intelligent Signaling simulation With 33% of the routers as reactionary, the dissemination of information amongst routers is focused, and thus spreads more quickly; resulting in a greater percentage of the network quarantined.

In the more realistic simulation the impact of the focused signaling is pronounced as the sub-network of routers quickly learns of the worm. In comparison to Figure 12, the sharp increase in the number of learned routers both occurs shortly after time 200. However, the intelligent signaling in the most recent simulation results in the set of reacting routers spreading this knowledge slightly faster than in the fire-with-fire approach.

The improvement in the share of the network that was successfully quarantined underscores the importance of speed. The only difference between the two realistic simulations in Figure 12 and Figure 13 is the means by which the routers communicate. The more efficient communication resulted in roughly a 7% increase in network protection (see Table 9). Furthermore, this paradigm incorporated an attempt at avoiding the routers acting blindly and continually sending signals by only sending responses to malicious code. Thus, the routers will stop attempting to alert others when the threats stop occurring. Additional improvements to such a signaling scheme may include observing the number of signals received to assume how long the invasion has been going on and whether or not sending additional signals would be beneficial.

Table 9: Maximum infestation percentages for the realistic simulations

Max Infested(R)	Drop	Drop w/ FwF	Drop w/ Sig
33% Reactors	97.0%	79.3%	72.5%
67% Reactors	70.7%	46.8%	43.4%
100% Reactors	43.9%	19.5%	17.3%

The most important trend in Table 9 data is the relationship between the number of reacting routers and the portion of the network that was successfully quarantined. A measure of the efficiency of the participation levels can be quantified by a ratio of the percentage of responding routers over the percent of the network that had been successfully quarantined (note that this is not the value portrayed in Table 9; infection percent was presented above). The ratios for one-third, two-third and complete reactionary routers are: 83.33%, 84.48% and 82.7%, respectively. Thus, adding

additional reacting routers increases the overall level of quarantine that may be achieved, but there appear to be diminishing returns on the investment. The exact relationship between reacting routers and the efficiency ratios are outside the scope of this research and is left for further study. Despite the lack of information regarding the best percentage of reactionary routers that are needed to achieve the highest ratio of efficiency, it should be noted that a complete network of reactionary routers under the realistic simulation parameter set was successful in quarantining over 84% of the network.

Signaling has shown to be one possible solution to actively alert peer routers of the presence of malicious code. Quite surprisingly, the speed at which signaling is sent through out the network is not as important as one might suspect when the speed of the routers recognition is sufficiently fast. On the contrary, if the recognition is not that quick, then signaling allows for the set of first responding routers to alert the group. However, even after all routers have been alerted, there will exist unprotected paths so long as there are non-reacting routers present in the network. Lastly, the intelligent signaling has shown promise by focusing the signals to known reactors and gaining an additional 7% of network protection. The proportion of the network that may be quarantined is directly related to the number of reacting routers. The goal is to find a way to get ahead of the worm's progression and allow other locations to prepare a defense when the recognition is not sufficiently faster than the worm.

CHAPTER 4 - SUMMARY & CONCLUSIONS

This paper has evaluated some possible means to defend against an attack by way of self-propagating code. The two primary categories of defense investigated were traffic throttling and content filtering. Traffic throttling was shown to be effective only in delaying the amount of time before the network succumbs to the intruder. Content filtering assumes that recognition is possible (among other assumptions) and then attempts to determine if such a system is capable of stopping an invasion and what degree of participation is required to do so.

Initial tests involved basic content filtering that did not attempt to share or spread information gained by the routers. These experiments successfully reduced the level of infestation achieved by the worm and greatly increased the time required for the network to fall victim to the code. Homeostasis was delayed by nearly 10,000% with less than 20% of the nodes becoming infected when two-thirds of the routers in the network were allowed to react. A complete network of reacting routers was shown to be incredibly resilient to an invasion with the worm maximizing its penetration at about one-quarter of one percent near time-step 236 (roughly half the time required for the base case to converge).

Then a slightly more reasonable set of simulations parameters were used to better estimate the performance of technology in the near future. These simulations showed that the routers attempted to stop the spread, but just were not fast enough. There was an early jump in the number of routers that were aware, then it leveled off as the worm had hopped to the far reaches of the network. It was in these initial outstretches that the worm mounted an offensive that the routers were simply not fast enough to stop.

The extreme basic content filtering test also revealed information regarding the minimum level of involvement from the reactionary routers needed to suppress or manage the invasion. The one-third and two-thirds router participation needed 98% and 58% of its pool of reactionary routers in order to achieve its level of protection. These percentages equate respectively to 38% and 39% of the total routers within the network. However, when the network is comprised of nothing but reactionary routers, the actual number of routers that are actively involved in the defense (those routers that learn of the worm) falls dramatically to about 2½%. This implies that the actual number of participating routers is not as important as where the reactionary routers are in respect to the source and the initial population of infected nodes. The luxury that is gained by having an entire network of reactionary nodes is that the network may defend itself appropriately at any point. The network does not contain an unprotected path between any two nodes.

Observations regarding location importance and the lack of the defense system to stop the worm in the more realistic setting resulted in attempts to spread information about the attack as quickly as possible throughout the reacting routers. The goal being to get a defense activated as quickly as possible by having protection in the right place at the right time.

In an attempt to quickly disseminate information amongst the routers about the threat, a system of signaling was based on the adage: fight fire with fire. The routers' signaling was completely random and directed at any other router in the network regardless of its abilities. The extreme simulations allowed the router to signal on every time step once it became learned while the realistic simulations placed a frequency

restriction on the router (equal to the worm for these experiments). The extreme scenarios saw some improvements in the ability to quarantine portions of the network and the realistic situations were able to stop some of the worm for the first time. Signaling showed promise, but the current system had obvious errors: it did not focus its work on known reactionary routers and the routers would send out signals for infinity. Both points result in a less network utilization because there are many useless packets and wasted time and effort as a result of the signaling policy. To alleviate a portion of this burden, a better signaling system was designed and simulated.

Intelligent signaling was devised in an attempt to mitigate the drain on network resources without impacting the increase in defense. This system had two important changes from the first iteration of signaling, fire-with-fire. In this new approach, the router only sends signals to other routers in the network that are known to be reactionary, thus eliminating wasting signals destined for routers that have no way of using the signaling information productively. Secondly, the routers only produce signals at the time that it drops a packet, ensuring that network resources are not further strained than in the event of a defenseless network. Further, this mechanism scales well with the invasion of the worm: as the worm infects more nodes and there are accordingly more malicious packets, the routers will produce more signals that help to prevent the outbreak.

The streamlined signaling yielded comparable results as those obtained when the network was effectively flooded with signals for the extreme case and offered a 7% increase in quarantined nodes in the realistic situation. Thus, there is a benefit from actively sharing information between routers. However, the sharing has a limited value

when the router alone can recognize the attack quickly. The relationship between the speed of recognition and signaling was not addressed and is left as further research.

This research concludes that there is a theoretical point at which recognition and information sharing can successfully stop a worm from completely occupying a network. The level of protection is obviously directly related to percentage of routers that are capable of providing protection and if there are holes in the network, the worm will eventually find them. The goal is to provide information as quickly and efficiently as possible so that those areas with the option of quarantine can take appropriate action and set up defenses. A focused distribution of knowledge does provide some additional level of security and results can be obtained with as little as one-third participation. For when the time comes technology has advanced to the point that malicious data can be positively identified, there should be a reliable system of defense to utilize the technology as efficiently as possible.

CHAPTER 5 - FUTURE WORK & ATTACKING THE SIMULATION

The work performed for this paper has been enlightening and with merit, but that is not to say that it is complete. There are areas in which the effect and nature of the network architecture and proliferation of malicious code are not completely understood. Some of these aspects have been mentioned throughout the paper and are repeated here with a bit more clarification, as well additional possible avenues of research that have yet to be discussed.

The structure used by the simulation is a strict tree structure such that all nodes have exactly one parent. There is, of course, an exception for the root node that has zero parents. This structure is not a true representation of the Internet, and it is impossible to properly characterize all private networks around the world in a single design. However, this basic structure does provide a reasonable approximation to the topology and basic functionality of the Internet. Examples of improvements that could be added to the simulation include the addition of peer connections and to vary the number of routers that are involved in transporting packets through the network. Further, specific routing protocols could be integrated as well. Nonetheless, the basic workings of the Internet are currently present and serve as a valid simulation and whatever pitfalls the worm encounters are also encountered by the signals; it is a fair simulation.

The current simulation only models worms similar to Slammer (single UDP packets that propagate at completely random destinations) [9,18]. Simulating the behavior of other worms and other infection strategies would help to strengthen the model and its claims.

Appendix A lists all the simulation parameters that can be tweaked to customize the simulation. Only a subset of these parameters were used in this research and further investigation on structure, speed, reaction, network utilization in transportation, etc. may be the focus of future studies.

The signaling system incorporated in the final set of experiments was better than the first iteration, but there is room for improvement. Knowledge about the number of worm packets that have been seen, the source of the malicious packets as well as the number of peer signals received and their sources could provide clues as to how to best alert other reactionary routers. Finally, one could base the signal destination from the information within the packet. It is possible that the source may be falsified, but the destination must be true since that is all the routers have to guide them. In such an instance, the simulation parameters allowing for priority channels for the signals could be utilized to add another factor of complexity to the problem.

APPENDIX – CONSTANTS FILE

```
#pragma once
using namespace std;
#define   VERBOSE   1
#define   EXCEL_DELIM    "\t"
#define   ADDR_DEFAULT   "DEFAULT"
/*************************************************************
  Overall Simulation Constants
    - MAX_AS_DEPTH  : Maximum to which the tree may grow.
          root is depth zero
    - TOTAL_ASES    : Total AS objects possible within the
          simulation
    - TOTAL_ROUTERS : Total Router objects that may exist
          in the simulation
    - TOTAL_NODES   : Total Node objects that may exist in
          the simulation
    - MAX_TIMER     : Total time steps that that simulation
          runs
    - TIMER_RECORD_INTERVAL : The approximate number of
          measurements that will be taken over
          the course of the simulation
*************************************************************/
#define   MAX_AS_DEPTH   5
#define   TOTAL_ASES     256
#define   TOTAL_ROUTERS  1024
#define   TOTAL_NODES    16384
#define   MAX_TIMER 2000
#define   TIMER_RECORD_INTERVAL    225
/*************************************************************
  Node Restrictions
    - PERCENT_INVULNERABLE    : The percentage of nodes
          that are invulnerable to infection
    - INITIAL_INFECTED        : The number of nodes to be
          randomly infected at time zero
*************************************************************/
#define   PERCENT_INVULNERABLE     .0
```

```
#define   INITIAL_INFECTED    1
/*********************************************************
  Router Restrictions
    - ROUTER__SCALING     : The number of routers may be
         inversely proportional to the depth ofthe AS.
         Nodes may have a direct relationship. This value
         determines the scaling factor in that
         relationship
    - ROUTER__PROC_DELAY: The base number of steps that a
         packet remains in the router queue. This delay is
         also scaled directly with the depth of the AS.
    - ROUTER__QUEUE_CAPACITY  : The maximum number of
         packets that a router may have in its queue
    - ROUTER__REACTION_TIME_CONSTANT     : The number of
         malicious packets that must be processed before
         the router "learns" of the signature. Expressed
         as an integer of packets
    - ROUTER__REACTION_TIME_PERCENT      : The number of
         malicious packets that must be processed before
         the router "learns" of the signature. Expressed
         as a decimal greater than zero. Determines the
         number of packets with respect to the number of
         children. This number is directly multiplied by
         the number of children to obtain that value.
         Percentage takes precedence over absolute value.
         Enter 0 or a negative number to use absolute
         numbers of packets.
    - ROUTER_THROTTLE__DELAY_FACTOR: Factor by which the a
         reating router will hold worm packets. IE: 2
         results in holding worm packets twice as long as
         regular packets
    - ROUTER_RESPOND_MIN_DEPTH: The minimum depth that
         routers will be able to respond
    - ROUTER_RESPOND_MAX_DEPTH: The maximum depth that
         routers will be able to respond
    - PERCENT_ROUTER_RESPOND  : The percentage of routers
         within the allowed reacting range (see previous
```

```
        two values) that will participate by being
        allowed to react to the invasion
    - SIGNAL__DELAY_FACTOR    : Decrease the queuing time
        for signal packets with respect to worm packets.
        IE: 2 would result in signals being queued for
        half the time as worm packets

    Boolean values determining how a router will respond:
    - ROUTER__TRAFFIC_THROTTLE
    - ROUTER__DROP_PACKETS
    - ROUTER__SIGNAL__RESPONSE     : Signal in response to
        a forwarded worm packet
    - ROUTER__SIGNAL__FIRE_WITH_FIRE    : Signal in a
        manner similar to a worm
    - SIGNAL_PROPAGATION_RATE : How often should the
        router send signals ?
    - SIGNAL__RANDOM     : Signal to random routers in the
        network
    - SIGNAL__RANDOM_INTELLIGENT   : Signal only within the
        reactionary routers
**********************************************************/
#define   ROUTER__SCALING     1
#define   ROUTER__PROC_DELAY  2
#define   ROUTER__QUEUE_CAPACITY   1000
#define   ROUTER__REACTION_TIME_CONSTANT     500
#define   ROUTER__REACTION_TIME_PERCENT 0
#define   ROUTER_THROTTLE__DELAY_FACTOR 2
#define   ROUTER_RESPOND_MIN_DEPTH 0
#define   ROUTER_RESPOND_MAX_DEPTH MAX_AS_DEPTH
#define   PERCENT_ROUTER_RESPOND   .33
#define   SIGNAL__DELAY_FACTOR     1
#define   ROUTER__TRAFFIC_THROTTLE 0
#define   ROUTER__DROP_PACKETS     1
#define   ROUTER__SIGNAL__RESPONSE 1
#define   SIGNAL_PROPAGATION_RATE  5
#define   ROUTER__SIGNAL__FIRE_WITH_FIRE     0
#define   SIGNAL__RANDOM 0
```

```
#define   SIGNAL__RANDOM_INTELLIGENT    1
/*************************************************************
  Worm restrictions
    - WORM_VALIDITY_FACTOR    : This is multiplied by the
          total number of nodes to restrict the frequency
          of finding a valid destination on the network
    - WORM_PROPAGATION_RATE   : The number of time step
          between each attempt by infected nodes to spread
          to another node
*************************************************************/
#define   WORM_VALIDITY_FACTOR     2
#define   WORM_PROPAGATION_RATE    5
/*************************************************************
  AS Restrictions
    - MIN_XXX  : The minimum number of objects that are
          generated for object XXX
    - MAX_XXX  : The value used to determine the maximum
          number of XXX objects. The range over which
          objects will be created is [MIN, MIN+RANGE]
    - PATH_INTERNAL_PERCENT   : Percentage of internal
          routers that will be process a packet when it
          reaches the AS
*************************************************************/
#define   MIN_AS_CHILDREN     8
#define   MAX_AS_CHILDREN_RANGE    4
#define   MIN_GATEWAY    5
#define   MAX_GATEWAY_RANGE   3
#define   MIN_INTERNAL   5
#define   MAX_INTERNAL_RANGE  3
#define   MIN_NODE  8
#define   MAX_NODE_RANGE 4
#define   PATH_INTERNAL_PERCENT    0.02
/*************************************************************
  Packet restrictions
    - MSG_ADDR_DELIMITER : The character used to separate
          sections of the address
    - PACKET__LIST_EMPTY : The value returned when the list
```

```
          of hops is empty
**********************************************************/
#define   MSG_ADDR_DELIMITER  '.'
#define   PACKET__LIST_EMPTY  -1
```

BIBLIOGRAPHY

[1] P. Akritidis, K. Anagnostakis, and E.P. Markatos: "Efficient Content-Based Detection of Zero-Day Worms," Proceedings of the International Conference on Communications (ICC 2005), Seoul, Korea. May 2005.

[2] T. Chen, J-M. Robert. "Worm Epidemics in High-Speed Networks," IEEE Computer. June 2004

[3] Cygwin Website. http://cygwin.com

[4] M. Eichin and J. Rochlis. With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988. Massachusetts Institute of Technology, Boston. 1988.

[5] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level traffic measurements from the Sprint IP backbone," IEEE Network Magazine. Nov 2003.

[6] Gnutella Website. http://www.gnutella.com

[7] J. Kurose and A. Ross, Computer Networking: A Top-Down Approach Featuring the Internet, 3rd Edition. Boston, Massachusetts, USA: Addison Wesley. 2005.

[8] P. Owezarski, "Does IPv6 improve the scalability of the Internet ?", Joint Internal Workshops on Interactive Distributed Multimedia Systems and Protocols for Multimedia Systems (IDMS/PROMS'2002), Coimbra, Portugal. November 2002.

[9] D. Moore, C. Shannon, G. Voelker and S. Savage. "Internet Quarantine: Requirements for Containing Self-Propagating Code," Proceedings of the 2003 IEEE Infocom Conference, San Francisco, CA. April 2003

[10] K. Patch. "Selective Shutdown Protects Nets," http://www.trnmag.com/Stories/2004/082504/Selective_shutdown_protects_nets_082504.html. August 2004.

[11] P. Roberts. "HP Shelves Virus Throttler". http://www.pcworld.com/news/article/0,aid,117531,00.asp. August 2004

[12] S. Savage. Personal communications. October 2004.

[13] E. Spafford, "The Internet Worm Program: An Analysis", Purdue Technical Report CSD-TR-823. Department of Computer Sciences Purdue University, West Lafayette, IN. November 1988.

[14] S. Staniford, V. Paxson and N. Weaver, "How to Own the Internet in Your Spare Time", Proceedings of the 11th USENIX Security Symposium, San Francisco, CA. August 2002.

[15] I. Stoica, R. Morris, D. Karger, M. Frans Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications," ACM SIGCOMM 2001, San Deigo, CA. August 2001. pp 149-160.

[16] L. Subramanian, S. Agarwal, J. Rexford and R. H. Katz, "Characterizing the Internet Hierarchy from Multiple Vantage Points," Proceedings IEEE INFOCOM 2002, June 2002. pp. 618-627.

[17] Worm Blog. http://www.wormblog.com

[18] D. Moore, V. Paxon, S. Savage, C. Shannon, and N. Weaver. "Inside the Slammer Worm," IEEE Security and Privacy, July 2003.

[19] K. Yokum, E. Eagle, J. Degesys, D. Becker, J. Chase, and A. Vahdat. "Toward Scaling Network Emulation using Topology Partitioning," in Proceedings of MASCOTS 2003. October 2003.

[20] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. "Network topology generators: Degree-based vs structural," In ACM SIGCOMM. August 2002

[21] S. Floyd and V. Paxon. "Difficulties in Simulating the Internet," IEEE/ACM Transactions on Networking Vol 9. August 2001.

[22] NS2 Website. http://www.isi.edu/nsnam/ns/

[23] SSFNet Website. http://www.ssfnet.org

[24] High-performance Computing & Simulation Research Lab Website, University of Florida. http://www.hcs.ufl.edu/

[25] C. Zou, W. Gong and D. Towsley. "Code Red Worm Propagation Modeling and Analysis," in 9th ACM Conference on Computer and Communication Security. November 2002.

[26] Z. Chen, L Gao, and K. Kwiat. "Modeling the Spread of Active Worms," in IEEE INFOCOM. 2003

[27] D. Moore, V. Paxon, S. Savage, C. Shannon, S. Saniford, and N. Weaver. "Inside the Slammer Worm," IEEE Security and Privacy, 1(4):33-39. July 2003.

[28] INDEX Group Website. http://lion.cs.uiuc.edu/overview.html

[29] C. Wang, J. C. Knight and M. C. Elder. On Computer Viral Propagation and the Effect of Immunization. Proceedings of 16th ACM Annual Computer Applications Conference. New Orleans, LA. 2000.

[30] Wikipedia website. "Autonomous System (Internet)" entry. http://en.wikipedia.org/wiki/Autonomous)system_%28Internet%29.

[31] T. Liston. "La Brea," http://hackbusters.net/LaBrea/.

[32] N. Weaver and V. Paxson. "A Worst-Case Worm," Proceedings Third Annual Workshop on Economics and Information Security (WEIS04). May 2004.