

CERIAS Tech Report 2004-35

IMPACT OF NETWORK DESIGN ON WORM PROPAGATION

by Brian Carrier and Sundararaman Jeyaraman and Sarah Sellke

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Impact of Network Design on Worm Propagation

Brian D. Carrier
carrier@cerias.purdue.edu
Department of
Computer Sciences
Purdue University

Sundararaman Jeyaraman
jsr@cs.purdue.edu
Department of
Computer Sciences
Purdue University

Sarah Sellke
ssellke@purdue.edu
School of Electrical and
Computer Engineering
Purdue University

Abstract

In this paper, we simulate the Code Red II and Nimda worms on different enterprise-scale networks to determine the impact that topology has on worm propagation. A corporate network can be designed to improve security and, as we show, to decrease the propagation rate of worms that use network scanning as a target discovery technique. We also examine the impact that LaBrea-like devices have on propagation rates and compare it to the impact of network topology.

1 Introduction

Due to their fast spreading nature and potential damage to the network infrastructure, worms have become a network security concern. Though there has been a good deal of work devoted to studying various aspects of worm propagation on the Internet, there has been little work that has examined worm propagation in a corporate enterprise network. Unlike the Internet, each company has complete control over the design of their internal network.

In this paper, we focus on the impact that network design has on worm propagation. Using simulations, we examined the propagation of a Code-Red-like and a Nimda-like worm through a typical enterprise network of approximately 10,000 hosts. Two different network topologies were simulated with each worm. We also simulated the impact that LaBrea-like devices would have on the worm propagation. In total, eight scenarios were simulated.

In this paper, we show that the propagation of physical network scanning worms, such as Code-Red, can be mitigated by the introduction of internal firewalls and LaBrea-like devices. When faced with a worm that uses both physical and logical networks (e.g. an e-mail network) for propagation, such as Nimda, these mechanisms are found to be ineffective and additional logical network defensive techniques are needed. In addition to providing results on the impact of network design, we think this is the first work to simulate a worm like Nimda that uses both e-mail and network scanning as an infection mechanism.

In Section 2 of this paper, we examine the background work including previous work on worm simulations and modeling, a description of the worms that we simulated, and an overview of network defense mechanisms. Section 3 describes the two network topologies that we simulated and Section 4 describes the network and e-mail simulator that we built. Section 5 provides the simulation results, Section 6 provides future work, and Section 7 concludes the paper.

2 Background and Previous Work

2.1 Worm Descriptions

In this section, we give an overview of the two worms that were simulated. Both worms were released on the Internet and caused widespread infections. They were chosen because they have different activation, or infection, techniques and

because they were successful in infecting millions of systems [5].

2.1.1 Code Red II

The CodeRed II worm exploits a buffer-overflow vulnerability in IIS servers running on Windows 2000 [9]. An infected server will scan for and infect other vulnerable servers. When a server is infected, the worm code remains dormant for one day and then starts to scan for vulnerable servers. When the worm generates an IP addresses to scan, it chooses 50% from the same class-A, 37.5% from the same class-B, and 12.5% from a random network.

2.1.2 Nimda

The Nimda worm affects Microsoft Windows 9x/ME, NT 4.0 and 2000 machines [3]. Nimda is unique because it uses four different propagation techniques to infect other systems. It will scan for and infect IIS servers that are vulnerable to a Unicode and escaped character decoding vulnerability. It will search the system for e-mail addresses and send an e-mail to the address with the worm as an attachment. Nimda will also search for local files with an HTM, HTML and ASP extension and append itself in a script that will exploit vulnerable versions of Internet Explorer. Lastly, Nimda will search for open network shares and copy a malicious DLL file to them. When a host is infected, it immediately attempts to use all four infection vectors. When the worm generates an IP addresses to scan, it chooses 50% from the same class-B, 25% from the same class-A, and 25% from a random network.

2.2 Worm Modeling and Simulation

Traditionally, researchers have studied the spread of computer viruses and worms by using epidemiological models that were developed to study the spread of infectious biological diseases [6]. Kephart et al., were the first to apply epidemiological models to model propagation of computer viruses in a series of studies [16, 15, 14]. Staniford et al. modeled the spread

of the CodeRed worm using the classical epidemic model [22] and Zou et al. modeled Code Red with the effect of human countermeasures and network congestion [27].

Chen et al. presented a model called the Analytical Active Worm Propagation (AAWP) model that uses a discrete time model and deterministic approximations to model network scanning [4]. This model can also examine worms that favor the local network when scanning. These models were not used for our work because they focus on the spread of worms in the Internet, take only network scanning worms into account, or cannot easily incorporate filters and firewalls.

Realistic mathematical models are the best way to characterize and study computer worm propagation, but they are usually hard or even impossible to create [24]. Similarly, creating realistic testbeds is also difficult because of the size requirements. Therefore, simulation is an effective technique that is used to understand and study worm propagation.

Wagner et al. [24] developed a worm simulator and documented their experiences with it. Two popular simulation frameworks that can model worms are Ssfnet [21] and EASEL [7]. N. Weaver also wrote a small, abstract simulator of a Warhol worm's spread [25]. The NWS simulator [8] is a simple Perl simulator that examines connectivity between hosts and does not take latency and bandwidth into consideration.

2.3 Defense Techniques

2.3.1 Containment

Moore et al. [18] simulated firewalls, content filters, and automated routing blacklists on the Internet to contain a worm outbreak. They find that content filtering can be more effective than address blacklisting. Zou et al. [28] analyzed an Internet-scale dynamic quarantine.

The previous two containment strategies were for the Internet, but Staniford et al. [23] examined the effectiveness of containment strategies in enterprise networks by identifying and blocking suspicious IP behavior. In our work, we

study if the static design of enterprise network topology can be effective in mitigating worm propagation.

2.3.2 Network Topology

Recent work has been conducted to identify a relationship between worm propagation and network topologies. Briesemeister et al. [2] study the spread of computer worms in artificially created scale-free networks (power-law networks), which the Internet is. They report that some scale-free network topologies are inherently more defensible than others against rapidly spreading computer worms.

Newman et al. [19] explore the spread of email worms in the “logical network” created by e-mail connectivity. The connectivity graph from e-mail address books is semi-directed and shows a strongly connected giant component. The authors show that the outbreak size can be significantly reduced if 10% of nodes are removed from the giant component.

2.3.3 Secure Network Design

There is little formal work specifically on secure network design, but there are guidelines that are offered by network companies and general security guidelines that can be easily applied [10] [12] [20]. The basic premise is to identify the value and the risk associated with each system, or asset. Systems with similar value and functionality can be grouped together and extra security measures placed around them.

In the case of an internal corporate network, this typically involves identifying the servers and desktop systems that have the most valuable information. The information could be financial records, customer information, and other IP. For example, with a secure design the financial department may be a separate subnet from the factory workers and firewalls prevent access to the financial subnet except for key services from specific subnets.

2.3.4 LaBrea

The LaBrea tool [17] is a simple program that Tom Liston designed after the outbreak of Code Red to reduce a worm’s scanning rate. It is a service that listens on the network and waits for an ARP request that is not being responded to. When this occurs, it assumes that the IP address is not being used and it responds and establishes a TCP connection. It will not close the connection or respond and will wait until the sender times out. The sender will then use exponential back off to resend the packet, which may repeat for up to two minutes. Although, the worm may give up earlier than that.

Chen et al. [4] used their AAWP model to evaluate the performance of LaBrea in mitigating worm propagation through the Internet. They found that the Internet needs at least 2^{18} LaBrea hosts to effectively defend against active worms.

3 Network Designs

This section describes the network topologies that we used to simulate the worm propagation. The goal was to simulate a typical 10,000 person enterprise network that has not been designed for security and a similar network that has.

To determine the number of servers that typically exist in a 10,000 person company, we asked colleagues who have done security and network consulting for companies of this size because no reference material could be found with this information. Based on the worms that we were simulating, we were concerned with the number of web servers and the number of desktops.

The enterprise network that we simulated had 400 web-based applications and a production environment with two applications per web server. Each application may have one QA and one staging web server. To determine where to put the production servers, we used research that shows that 80% of traffic leaves a group or subnet and is destined for a remote server and 20% of traffic stays in the local network [11]. Therefore, we put the application servers in central server farm and placed two Intranet servers in each subnet.

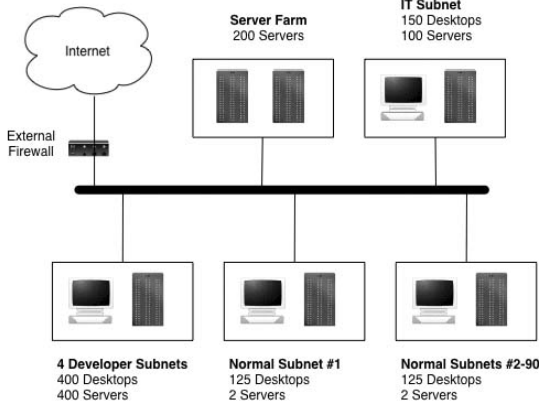


Figure 1: The flat network topology that was used to simulate the worm propagation.

The simulated network had a total of 96 class-C subnets containing 11,800 desktop Microsoft Windows systems and 880 Microsoft IIS web servers. The topology of the typical network, also called the flat network, can be seen in Figure 1. In this network, each system had connectivity to every other system. There was one subnet for the server farm with 200 production IIS servers. There were four subnets for the software developers and each of these had 100 desktop and 100 IIS servers. The IT subnet had 150 desktop systems and 100 IIS servers. The remaining 90 subnets were for “normal” desktop users and each had 125 desktops and 2 IIS servers. The flat network had no internal firewalls.

This network was made more secure by adding firewalls to each of the subnets. The following filtering rules were used:

- All subnets can send e-mail to all subnets
- All subnets have outbound HTTP access to the Internet.
- The server farm subnet allows inbound HTTP access from all subnets.
- All “normal” subnets allow inbound HTTP access from only the IT subnet
- All developer subnets allow inbound HTTP access from only the IT subnet and other developer subnets.

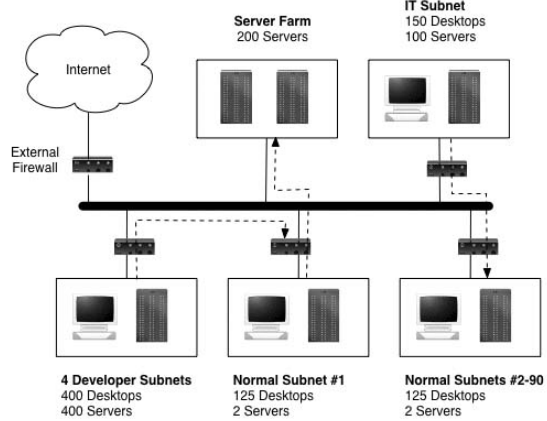


Figure 2: The filtered network topology that was used to simulate the worm propagation.

- The IT subnet denies inbound HTTP access from all subnets.

We can see the the filtered network in Figure 2. We see that the developer network cannot connect to servers in the normal subnets, but that IT systems can. Any system can make connections to the server farm.

Our simulations also included two scenarios that added LaBrea-type hosts to the flat network design. The first scenario filled the unused IP addresses in the server farm, developer network, and normal subnets with LaBrea hosts. The second scenario created 50 new class-C subnets filled with LaBrea Hosts.

For our simulations, we assumed that there was a firewall protecting the enterprise network from the Internet and that it blocked incoming HTTP traffic. We assumed that the initial network infection was from an employee’s laptop that was infected while at home. Therefore, the infection will start in one of the developer, IT, or normal subnets.

4 Simulator Design

4.1 General Design

For our simulations, we modified Bruce Ediger’s NWS [8] simulator. It was chosen because it was basic and primarily concerned with the connectivity of networks and not the details of delay

and throughput. Ultimately, it was too basic and we had to add support for subnets, routing, and filtering.

The Perl-based simulator initializes by creating hosts and network connectivity. The simulator works in cycles and each host is examined during a cycle. If the worm has infected a host, then the host will perform one scan during the cycle. If a host has not been infected, then it will do nothing during the cycle. When a host needs to communicate with another host, it sends a message, which is the equivalent to a network packet. The message payload is code that the recipient will execute if they accept the message. In our case, the payload is the worm code. Network filters can block the message from being delivered.

4.2 Simulating Network Propagation

This section will describe how we simulated the network scanning worms. The first step was to create the network topology using three types of hosts: servers, desktops, and LaBrea. All of the hosts started with no resident code. When the simulation started, a host in a specified subnet was infected by adding the worm code to it.

The worm code used the worm's specific scanning algorithm to identify other servers send messages to. If the receiver of the message was vulnerable and not already infected then it would become infected. To simulate the time required to download the code and infect the victim, both the sender and victim would sleep for the next 6 cycles. When a LaBrea host was infected, it would cause the sender to sleep for 120 cycles. This was to simulate the sender not being able to close its network connection. Both of these values were arbitrarily chosen, but should not have significant changes on the results. Filters were added to the network topology when the filtered network was simulated.

4.3 Simulating E-mail Propagation

To simulate e-mail propagation, each desktop host was considered to have a user that would check email from time to time. In the simulator,

e-mail was sent using the same message infrastructure that was used to pass server traffic, except it was sent on port 25. Based on the general design of email and the goals of the simulation, we were presented with the following major challenges:

- **Modeling e-mail network topology:** E-mail worms propagate using a logical network based on address book entries and not a physical network topology.
- **Modeling e-mail checking patterns:** E-mail worms, in general, infect the host only after the user manually opens the attachment and not when the message is first received.

In our simulation, we assumed that a user opened all of his new e-mail when the e-mail was checked. If one of the new e-mails contained a worm message, the host became infected and the worm code was be executed. We did not model re-infections i.e., if an infected system received a second worm than nothing happens. We assumed that address books would not change through the duration of the attack. This was not unreasonable considering that our simulation time frame was less than an hour. In the following sections, we describe how we approached the aforementioned challenges. The terms *e-mail graph*, *e-mail topology* and *users*, *nodes* are interchangeably used in the following discussion.

4.3.1 E-mail Network Topology

To the best of our knowledge, only two studies have examined e-mail network topology. Newman et al. [19] used the address books of users at a university campus to build a graph model. The resulting graph had a giant strongly connected component and was semi-directed. The degrees of the graph nodes were found to decay following either a simple or a stretched exponential distribution.

Zou et al.[26] used three models for generating e-mail networks: a random-graph model, a small-world model, and a power-law model. Based on their observation of Yahoo! groups

mailing list sizes, the authors concluded that a power-law model was the most accurate model. In a power-law model, the distribution of the degrees obey a power-law distribution.

We hypothesize that in a typical organization that is structurally divided into departments (like the one we simulated), the following assertions hold:

1. E-mail address books will have most of the entries filled with users in the same department.
2. For any two users Alice and Bob, if Alice has Bob in her address book, Bob is likely to have Alice in his address book.

Assertion 1 implies that the e-mail graph derived from the address books should exhibit small-world characteristics with clustering between users within the same department. Assertion 2 implies that most of the edges in the resulting graph should be bi-directional. Based on our own hypothesizes and the observations by Zou et al., it was clear that we needed an e-mail graph that exhibited small-world characteristics with mostly bi-directional edges and node degrees that followed a power-law distribution.

An algorithm by Jin et al. [13] was used to generate such graphs. It includes a value p , $0 \leq p \leq 1$, called the *local probability*, which is used to determine the proportion of connections to local nodes. The algorithm is as follows.

1. Assign a degree d_i to each user i , $1 \leq i \leq N$, such that d_i follows the power-law distribution with α as the power-law exponent and N is the number of e-mail users in the enterprise network.
2. Create local connections by connecting each user i to $p * d_i$ users in the same subnet.
3. Create remote connections by connecting each user i to $(1 - p) * d_i$ users in other subnets.

Jin et al. observed that a network with the characteristics that we desire can be created with small values of the power-law exponent α

($1 \leq \alpha \leq 2$) and moderate values of the local probability p ($0.5 \leq p \leq 0.7$). In our simulation, we used the values $\alpha = 1.52$ and $p = 0.7$ to generate the e-mail network.

4.3.2 E-mail Checking Frequencies

To model user e-mail checking patterns, we used the work by Zou et al. Note that this pattern is based on when a user is in front of her computer and opens the e-mail and not based on the regular interval that the client uses to check for new e-mail.

The *e-mail checking interval* between consecutive e-mail checks by a user i is defined by a variable T_i , $i = 1, \dots, N$. The variable T_i is assumed to be an exponentially distributed random variable with a mean $E[T_i]$. We also assume that $E[T_i]$ is a random variable, denoted as T . Since the number of users N is large and user habits are typically independent of each other, T can be safely assumed to approximate to a normal distribution (central-limit theorem) i.e., $T \sim N(\mu_T, \sigma_T^2)$. In our simulations, we use $T \sim N(8, 100)$. Though these numbers are arbitrary, any change in them is unlikely to significantly affect the results.

4.3.3 Nimda Simulation Implementation

When the simulator is initializing, each desktop host is assigned an address book degree and a random variable for its email checking frequency. After all hosts have been created, we use the power-law distribution to build the address book for each desktop host.

When an e-mail message is sent to a desktop, it is placed in the receiver's queue. At each cycle after receiving the e-mail, the host produces a exponentially distributed random variable to determine if it will open the email in that cycle. Once the desktop "opens" the email and executes the virus message, it sends virus messages to the other desktops in its address book and starts to scan for vulnerable servers. The other desktops that the message is sent to will not become infected until they randomly remove it from their queue.

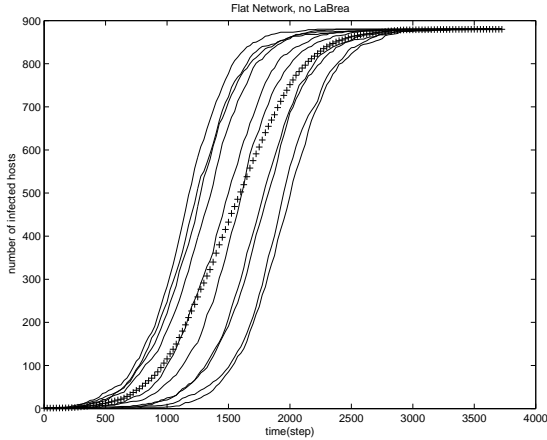


Figure 3: 10 Rounds of Code Red II propagation in the flat network model. The star line is the average of the rounds.

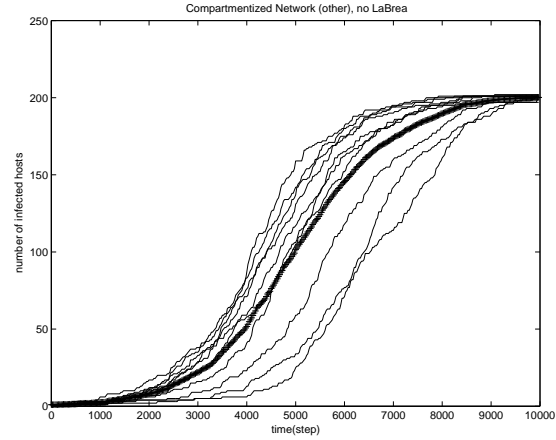


Figure 4: 10 Rounds of Code Red II propagation in the filtered network starting in a normal subnet. The star line is the average of the rounds.

5 Simulation Results

5.1 Code Red II Worm Results

We simulated four scenarios for the Code Red II worm in the flat and filtered networks with no LaBrea hosts. For each scenario, the simulation was performed 10 times and the average was calculated. We will first show the results of each scenario and then compare the results. Note that Code Red infects only IIS servers, so we are not concerned about the total number of desktops. As a reference to the real world, it was observed that servers infected by Code Red were conducting roughly two scans per second [4] and therefore we can assume that two simulator cycles occur per second. This is an estimate, but any change in the value should not change the relative changes between each scenario.

The first scenario was to simulate the worm on the flat network. The results, shown in Figure 3, show the growth of infected servers and it has the typical S shape. 10% of the vulnerable servers were infected in about 750 to 1500 cycles. On average, 10% of the vulnerable servers were infected in 950 cycles and all 880 servers were infected in 2750 cycles.

The next three scenarios used the filtered network topology. The difference in each scenario is where the worm started from: developer subnet, IT subnet, or a normal subnet. The location will

affect how many other servers can be infected.

The second scenario was when the infection started in a normal user subnet. These results can be seen in Figure 4. The normal subnets had access to the two servers in their subnet and the 200 servers in the server farm, which is 23% of the total servers. In this scenario, 10% of all servers were infected between 4000 to 6000 cycles. On average, it took 4750 cycles for 10% of the servers to be infected.

The third scenario was when the infection started in one of the developer subnets. The results can be seen in Figure 5. A server in a developer subnet had access to 400 servers in developer subnets and the server farm for a total of 600 servers, or 68% of the total network. In this scenario, 10% of all servers were infected between 1000 to 3500 cycles. On average, it took 1900 cycles to infect 10% of the servers.

The fourth scenario was when the infection started in the IT network, which had access to all servers. The results can be found in Figure 6. In this scenario, 10% of all servers were infected between 1200 to 2000 cycles. It took 1550 cycles on average to infect 10% of the servers. This scenario had many more systems that it could infect and the servers in the normal subnets could be reached only from the infected IT servers. At the end of 10,000 cycles, only 4 out of 10 rounds had infected more than 800 of the 880 servers.

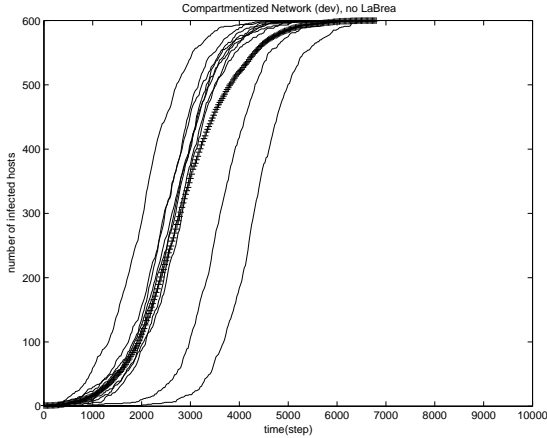


Figure 5: 10 rounds of Code Red II propagation in the filtered network starting in the developer’s subnet. The star line is the average of the rounds.

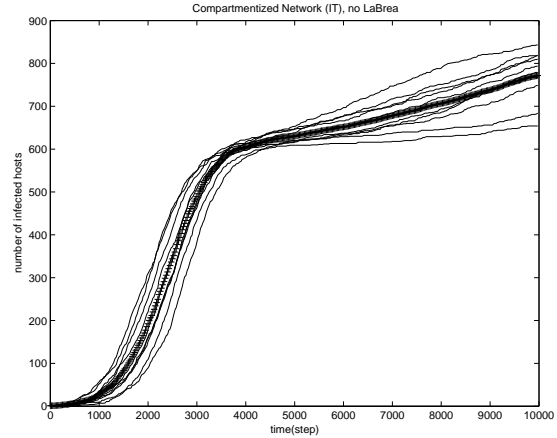


Figure 6: 10 rounds of Code Red II propagation in the filtered network starting in the IT subnet. The star line is the average of the rounds.

Figure 7 shows a comparison of the propagation rates of the flat network and each of the filtered network scenarios. If the 10% infection times from the three filtered network scenarios are averaged and compared to the times in the flat network, a 188% increase in the propagation rate can be seen.

The total infection from the normal user subnets is much smaller because it had access to only the subnet servers and the server farm. Unfortunately, with Code Red II, it is unlikely that a normal user would be infected by the Code Red II worm because it affects IIS servers. An infection seems more likely to start from the IT or developers network where it would be more common for them to run IIS.

5.2 Code Red II Worm with LaBrea

We next simulated the Code Red II worm on the flat network with LaBrea hosts installed. The goal was to determine if we could slow the propagation rate while not having to determine complex network topologies and firewall rules. When a LaBrea host was infected, it caused the infecting host to sleep for 120 cycles, or roughly 60 seconds. There were two scenarios that were simulated. Their results are presented and followed by a comparison to the results for the flat and

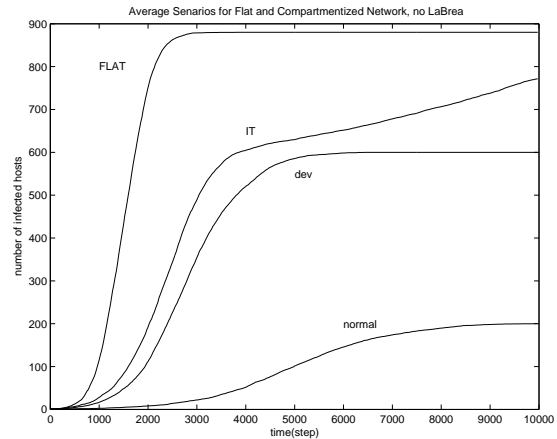


Figure 7: Comparison of the average Code Red II propagations in the flat and filtered networks.

filtered networks without LaBrea.

The first LaBrea scenario had LaBrea hosts installed in the unused IP addresses of the server farm, developer, and other normal subnets. This added 11,795 LaBrea hosts to the existing network. The results can be found in Figure 8. On average, it took 8,450 cycles to infect 10% and 23,750 cycles to infect 95% of the servers. In two of the rounds, not all of the 880 servers were infected after 25,000 cycles. In these cases, it took a long time for the systems that were initially infected to infect other servers because they were always getting stuck with a LaBrea host.

The second scenario added 50 subnets of 255

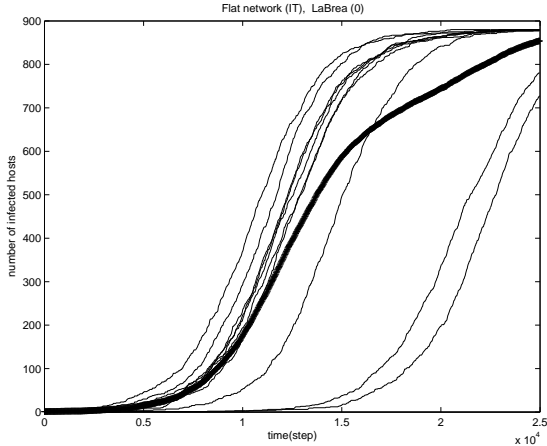


Figure 8: 10 rounds of Code Red II propagation in the flat network with LaBrea hosts using the unused IP addresses in each subnet. The star line is the average of the rounds.

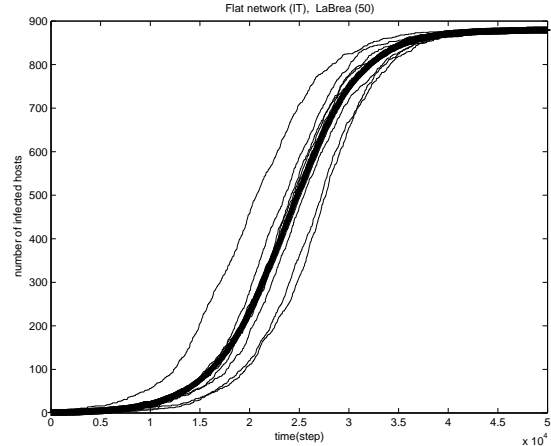


Figure 9: 10 rounds of Code Red II propagation in the flat network with 50 subnets full of LaBrea hosts. The star line is the average of the rounds.

LaBrea hosts to the flat network topology, for a total of 12,750 LaBrea hosts. The number 50 is arbitrary and has no special meaning. The results of this simulation can be seen in Figure 9. It took between 12,500 and 17,500 cycles to infect 10% of the servers. On average, it took 15,650 cycles to infect 10% and 35,000 cycles to infect 95% of the servers. This figure shows that it took 85% longer to infect 10% of the systems when using the LaBrea subnets instead of filling in unused hosts in existing subnets, even though only 8% more LaBrea hosts were used.

To compare the Code Red II results, we graphed all of the simulation results, as shown in Figure 10. You can see how the LaBrea scenarios had a much smaller propagation rate than the flat network and the filtered network.

A summary of all Cod Red simulations can be found in Table 1. We conclude that LaBrea is a powerful tool to slow down the propagation of worms, even when the enterprise network is not compartmentalized. Filling in the unused IP addresses in the flat network caused a 798% increase in time to reach 10% server infection and adding 50 LaBrea subnets caused a 1,547% increase.

Table 1: Summary of the cycles needed for 10% infection in different scenarios.

Network Design	Cycles until 10% Infection
Flat	950
Filtered - IT	1,550
Filtered - Developer	1,900
Filtered - Normal	4,750
Flat with unused IP LaBrea	8,450
Flat with 50 LaBrea subnets	15,650

5.3 Nimda Worm Results

In this section, we present our simulation results of the Nimda worm. The Nimda worm is a multi-vector worm that infects the desktops and servers, as described in Section 2.1. For this simulation, we used only the e-mail and network scanning infection routines. To determine how many seconds a cycle corresponded to, we looked for references that identified the observed scan rate of Nimda, but none could be found. Using the values used for the e-mail frequency checking, we think that six cycles occur per second. As noted for Code Red, the exact conversion value will not change the relative changes between topologies.

This worm takes much longer to simulate than

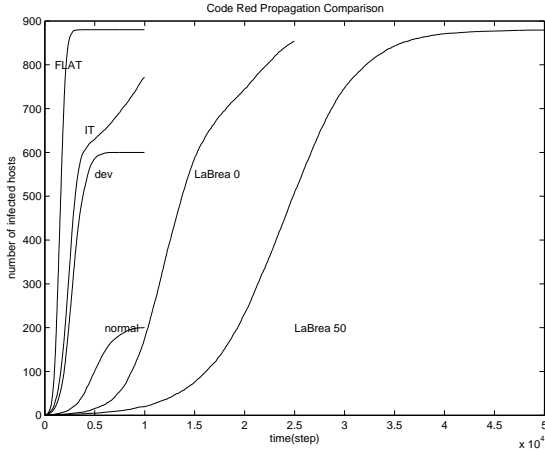


Figure 10: Comparison of the average Code Red II propagation in the flat, filtered, and LaBrea subnets. The shorter lines were not run for as many cycles because all reachable servers were infected.

Code Red II does because there are over 11,000 vulnerable hosts in our network. We simulated only five rounds of the infection in both the flat network and the filtered network and the results were averaged. The flat network results can be seen in Figure 11 and the filtered network results can be seen in Figure 12. The graphs show the desktop infection rate from e-mails, the server infection rate from scanning, and the total number of infections. As can be seen, these graphs are very similar and show that network topology did not have an impact on the propagation.

In many enterprise networks, the number of infected servers could be more important than the total number of infected hosts, so we examined the infection rate of only servers and ignored the large number of vulnerable desktops, as shown in Figure 13. We see that the filtered network slows the infection rate, but that all of the servers are infected in the end. This is because the virus can pass the firewalls via e-mail, so no server is immune from infection. In the flat network, it took 533 cycles to reach 10% server infection and 575 cycles in the filtered network. For 90% server infection, it took 1,275 cycles in the flat network and 3,325 cycles in the filtered network.

Figure 14 shows the total number of Nimda

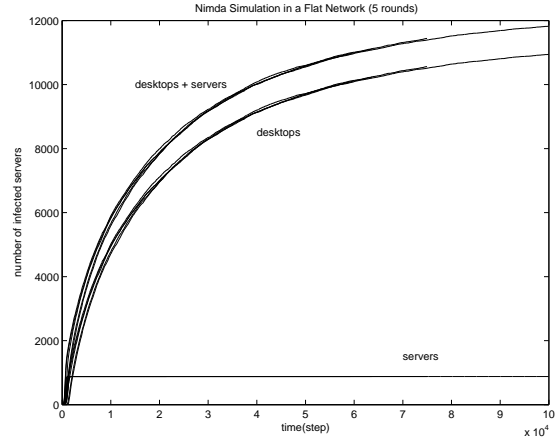


Figure 11: 5 rounds of Nimda propagation in the flat network.

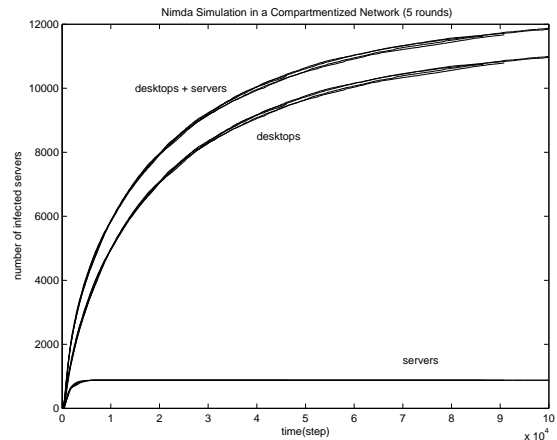


Figure 12: 5 rounds of Nimda propagation in the filtered network.

infections in the flat and filtered network simulations. The propagation is roughly the same because email was not being stopped at any subnet. To reduce the impact of multi-vector worms like Nimda, an email defense technique is also needed.

6 Future Work

Simulating worm propagation in an enterprise network is a new area and there are several other scenarios that should be considered. A basic change to our simulations would include the flat network with outbound filtering from the server farm subnet and filters within the server farm.

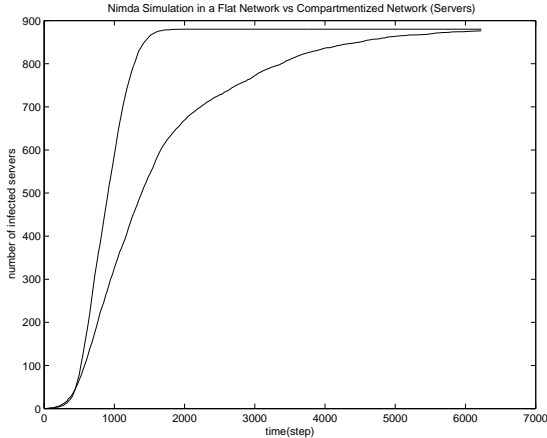


Figure 13: Comparison of average Nimda server propagation in flat and filtered network.

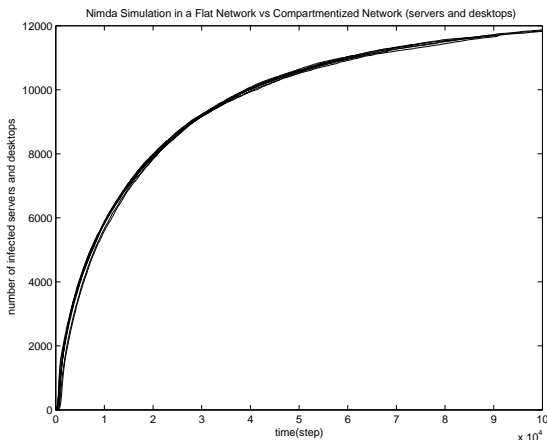


Figure 14: Comparison of total Nimda propagation in flat and filtered network.

This would decrease the infection rate once one of the servers was infected by a network scanning worm. Another topology may consider an enterprise network with multiple campuses, which may affect where the servers are located. Future work should also include more research into e-mail propagation and simulation techniques.

Research and commercial products are focusing on the detection and dynamic response to worm outbreaks in an enterprise [1, 23] and their effectiveness could be evaluated using simulators. For example, in our Code Red II simulation on the filtered network, we could have monitored the internal firewalls to detect a subnet that was sending lots of packets that were be-

ing dropped. The router for that subnet could have quarantined the network from the enterprise. The simulations could also account for systems being patched or new virus definitions that are installed while the e-mail virus is propagating.

7 Conclusion

In this paper, we have examined the impact that network topology has on worm propagation by simulating two widespread worms on different networks. An enterprise network can be more easily redesigned than the Internet and therefore this work could be used by network designers and companies when they examine the benefits of re-designing a network.

By simulating the Code Red II worm we were able to show that, on average, it took almost 3 times as long for our network with internal firewalls to reach 10% infection when compared with the same network without internal firewalls. On the other hand, by simulating the Nimda worm on the same network topologies, we were able to show that when a logical network, such as e-mail, is used by a worm, then static physical network defenses, such as firewalls, are ineffective. Dynamic defenses and logical network defenses may provide better results and it is likely that many worms in the future will use multiple attack vectors. We believe that this was the first work to simulate a worm using network scanning and e-mail as propagation techniques.

Using the Code Red II worm, we simulated the effect that the LaBrea network device has on slowing network infections. These results showed that LaBrea devices have a bigger impact than topology alone for network scanning worms. Further, the simulations showed better results from creating subnets filled with LaBrea hosts instead of filling in unused IP addresses in existing subnets. In practice, LaBrea systems may be easier to deploy in a large network because they do not require the network to be redesigned.

References

- [1] Arbor Networks. "Peakflow X." Available at: http://arbornetworks.com/products_x.php.
- [2] L. Briesemeister, P. Lincoln, and P. Porras. "Epidemic profiles and defense of scale-free networks," *In the Proceedings of the 2003 ACM Workshop on Rapid Malcode*, Washington DC, 2003.
- [3] "CERT Advisory CA-2001-26 Nimda Worm," <http://www.cert.org/advisories/CA-2001-26.html>.
- [4] Z. Chen, L. Gao, and K. Kwiat. "Modeling the Spread of Active Worms," *IEEE INFOCOMM*, 2003.
- [5] S. Costello. "'Nimda,' 'Code Red' still alive and crawling," *CNN*, May 8, 2002. Available at: <http://www.cnn.com/2002/TECH/internet/05/08/nimda.code.red.idg/>.
- [6] D.J. Daley and J. Gani, "Epidemic Modelling: An Introduction," *Cambridge University Press*, 1999.
- [7] EASEL web site. <http://www.cert.org/easel/>.
- [8] B. Ediger, "NWS Simulator". Available at: <http://www.users.qwest.net/~eballen1/nws>.
- [9] eEye Digital Security, "CodeRedII Worm Analysis," August 2001, <http://www.eeye.com/html/Research/Advisories/AL20010804.html>.
- [10] S. Garfinkel, E. Spafford, and A. Schwartz, "Practical UNIX and Internet Security - Second Edition," O'Reilly, 2003.
- [11] G. Haviland, "Designing High-Performance Campus Intranets with Multilayer Switching," Cisco Systems, July 2002. Available at: http://www.cisco.com/warp/public/cc/so/cuso/epso/entdes/high_wp.htm.
- [12] IBM, "IT Security: Creating value with comprehensive security solutions for public and justice agencies", 2001. Available at: <http://www-1.ibm.com/services/files/gsw1795f.pdf>.
- [13] S. Jin and A. Bestavros, "Small-World Internet Topologies: Possible Causes and Implications on Scalability of End-System Multicast," Computer Science Department, Boston University, Technical Report: BUCS-TR-2002-004.
- [14] J. Kephart, D. Chess, and S. White, "Computers and Epidemiology," *IEEE Spectrum*, 1993.
- [15] J. Kephart and S. White, "Directed-graph Epidemiological Models of Computer Viruses," *In Proceedings of the IEEE Symposium on Security and Privacy*, 1991.
- [16] J. Kephart and S. White, "Measuring and modeling Computer Virus Prevalence," *In Proceedings of the IEEE Symposium on Security and Privacy*, 1993.
- [17] T. Liston, "LaBrea". Available at: <http://labrea.sourceforge.net>.
- [18] D. Moore, C. Shannon, G. M. Voelker, and S. Savage, "Internet Quarantine: Requirements for Containing Self-Propagating Code," *In IEEE INFOCOMM*, 2003.
- [19] M. E. J. Newman, S. Forrest, and J. Balthrop, "Email networks and the spread of computer viruses," *Physical Review E*, 66, 2002. 035101.
- [20] D. Smith, "Improving Computer Security through Network Design," AUSCert, June 1997. Available at: <http://www.auscert.org.au/render.html?it=2249&cid=1920>.
- [21] Ssfnet web site. <http://www.ssfnet.org/>.
- [22] S. Staniford, V. Paxson, and N. Weaver, "How to Own the Internet in Your Spare Time," *In 11th Usenix Security Symposium*, San Francisco, August, 2002.

- [23] S. Staniford, "Containment of Scanning Worms in Enterprise networks," *Journal of Computer Security*, to appear, 2004.
- [24] A. Wagner, T. Dubendofer, B. Plattner, and R. Hiestand, "Experiences with Worm Propagation Simulations," *In Proceedings of the 2003 ACM workshop on Rapid Malcode*, 2003, Washington, DC, USA.
- [25] N. Weaver, "Warhol Worms: The Potential for Very Fast Internet Plagues," <http://www.cs.berkeley.edu/~nweaver/warhol.html>.
- [26] C.C. Zou, L. Gao, W. Gong, and D. Towsley, "Email virus propagation modeling and analysis," Department of Electrical and Computer Engineering, Univ. Massachusetts, Amherst, Technical Report: TR-CSE-03-04.
- [27] C.C. Zou, W. Gong, and D. Towsley, "Code Red Worm Propagation Modeling and Analysis," *In 9th ACM Symposium on Computer and Communication Security*, Washington DC, 2002.
- [28] C.C. Zou, W. Gong, and D. Towsley, "Worm propagation modeling and analysis under dynamic quarantine defense," *In Proceedings of the 2003 ACM workshop on Rapid Malcode*, 2003, Washington, DC, USA.