A Firewall Network System for Worm Defense in Enterprise Networks

Cliff C. Zou, Don Towsley, Weibo Gong
{czou,gong}@ecs.umass.edu, towsley@cs.umass.edu
Univ. Massachusetts, Amherst
Technical Report: TR-04-CSE-01

***Abstract*— From a security point of view, the Internet is too open. The central idea of a traditional "firewall" is to constrain service requests from the Internet to a local network. As an enterprise network becomes larger and more flexible, an Internet worm can easily find a way to enter it. Based on the "defense-in-depth" principle, we present a "Firewall Network System" for worm defense in an enterprise network that uses internal firewalls to divide the network into many isolated subnetworks. Computers in an enterprise network are classified as either clients or servers: all service requests sent to internal IP addresses of an enterprise network will be blocked by internal firewalls if they target non-server computers or servers that do not provide the corresponding service. In this way, the Firewall Network System removes most worm infection paths in an enterprise network, making worm detection much easier. All internal firewalls are designed to have the same set of firewall rules, which means the Firewall Network System is scalable and easily managed. In addition, we propose a five-level feedback worm defense strategy and present models of several worm defenses based on either active patching or quarantine.**

I. Introduction

Computer "worms" are programs that self-propagate across a network exploiting security or policy flaws in widely-used services [3]. From a security point of view, the Internet is too open: without the presence of security devices such as firewalls, any computer in the Internet can directly contact any other computer so long as the target computer has a global routable IP address. Because of this openness, computer worms have become one of the major threats to the Internet. Since 2001, several widely-spread worms, Code Red [8], Nimda [6], SQL Slammer [4], and Blaster [7], have repeatedly spread across the Internet and caused substantial damage.

Computer worms can spread throughout the Internet within hours, even minutes. For example, the SQL Slammer infected 90% of all vulnerable computers in the Internet within 10 minutes [4]. Such fast spreading worms motivate the need for an automatic worm defense system. However, building such a system in the global Internet is tremendously difficult due to the complexity of the Internet, the security and privacy issues in data sharing, and the cooperation required among all Internet communities. Hence, before we can build up such a global Internet worm defense system, there is a great need by organizations, especially enterprises, to first build up a worm defense system for their computer networks. In the following, we refer to the computer network of an organization as an "enterprise network".

Suppose a worm exploits a vulnerability on port x. When an infected host (source) attempts to infect a vulnerable host (target), the source needs to send corresponding TCP/UDP packets to the target on port x. If port x is a TCP port, then the source needs to first send a TCP SYN packet to the target on TCP port x to set up TCP connection, which is the case for Code Red and Blaster [8][7]; if port x is a UDP port, the source can directly send the exploiting code to the target on port x, as in the case of SQL Slammer [4]. We refer to such a network connection from the source to the target as a "*service request*" on TCP/UDP port x. A successful worm infection from an infected host (source) to a vulnerable host (target) requires that the source and target computers satisfy two conditions:

- The target computer accepts service requests on the vulnerable service port.
- The source computer can directly send attacking service requests to the target computer.

A "personal firewall" is a software product installed on a computer to preclude the first of these conditions: it disables all service ports on a computer unless the user explicitly permits traffic to certain service ports. However, it can only protect one computer and causes great trouble to ordinary computer users who know nothing about computer security. Most enterprise networks have installed "firewalls" between their local enterprise networks and the global Internet. A “firewall” explicitly permits certain incoming service requests from the Internet to a local network. It defends against a worm by ensuring that the second condition is never satisfied.

However, traditional firewalls are not sufficient nowadays to protect an enterprise network from worm attacks. As more enterprises implement wireless networking, Virtual Private Networks (VPN) and allow employees to work at home, it becomes harder to define the boundary of an enterprise network. At the same time, Internet worms become more complex and intelligent — they can easily find a way to go around a traditional firewall to infect computers in an enterprise network. For example, the Nimda worm [6] could traverse an enterprise traditional firewall by sending infectious emails. Therefore, boundary firewalls may not be able to block all worm infection attempts from the global Internet.

Traditional firewalls place a tight security check at the boundary of an enterprise network, while usually there is no such check for internal traffic. Thus they cannot defend an enterprise network once a worm finds a way to infect an internal computer. What is required is a “defense in depth”: in addition to traditional firewalls at the boundary of an enterprise network, firewalls should be distributed inside the enterprise network to restrict access among internal computers as well.

Briefly speaking, our idea is to introduce firewalls to prevent the second condition of a successful worm infection from being satisfied within an enterprise network. We refer to such a worm defense system as a *Firewall Network System*. We place firewalls on physical links of an enterprise network, dividing the network into isolated *subnetworks*. Henceforth, the traditional firewall found at the boundary of an enterprise network is referred to as the *boundary firewall*; firewalls within an enterprise network as *internal firewalls*.

The most important theme of our defense system is that we classify all computers in the enterprise network using the traditional “*client/server*” networking mode (computers in a peer-to-peer system are both clients and servers). All internal firewalls in the Firewall Network System deploy explicit access policies to allow predefined service requests (defined by source IP, destination IP, destination port) to pass through. Under this defense, all service requests sent to internal IP addresses of an enterprise network are blocked by internal firewalls if they target non-server computers or servers that do not provide the corresponding service to the sources. In this way, the Firewall Network System removes most worm infection paths in an enterprise network and also makes worm detection much easier.

Another important theme of our defense system is that we solve the system management issue by requiring all internal firewalls to have the same set of firewall rules. This makes the Firewall Network System scalable to large enterprise networks and easily managed by a central firewall management console. In addition, we propose a five-level feedback worm defense strategy based on the “feedback quarantine” principle from epidemic disease control [28] and the five-level US “Homeland Security Advisory System” [2]. In this way, the Firewall Network System can take appropriate (cost-effective) defense actions under different situations.

It should be noted that the Firewall Network System is designed for defense against “worms”, not “email viruses” such as recent “SoBig” [19] and “MyDoom” [20]. An email virus needs a user to execute virus attachment of an email and then the virus compromises the user’s computer without requiring the computer to have any vulnerability. On the other hand, a worm does not need human interference to propagate and compromises computers only if they have the corresponding vulnerability. Email viruses are different from worms in their propagation and infection mechanisms. However, the Firewall Network System can help our defense against email viruses (explained in Section IV).

The rest of this paper is organized as follows. Section II surveys related work. Section III presents the architecture of the Firewall Network System. The major principles in designing the Firewall Network System are introduced in Section IV. Section V discusses worm detection and defense issues and proposes a five-level feedback defense strategy. Section VI presents three worm propagation models based on different defense strategies and the corresponding experiments are presented in Section VII. In the end, Section VIII concludes this paper.

II. Related Work

The “defense in depth” is not a new concept; the NSA presented the concept of “defense in depth” in its security recommendation guides [13] in the context of a three-layer defense: “people, technology, and operations”. People have thought about using multiple layers of firewalls to protect an enterprise network for a long time. Currently many enterprises have set up additional firewalls to protect important servers inside their networks. J. Snyder studied the issues of “defense in depth” by pushing firewalls inside an enterprise network [9][10]. However, his studies did not present any new technology and concentrated mainly on user authentication and protection of servers. In addition, he did

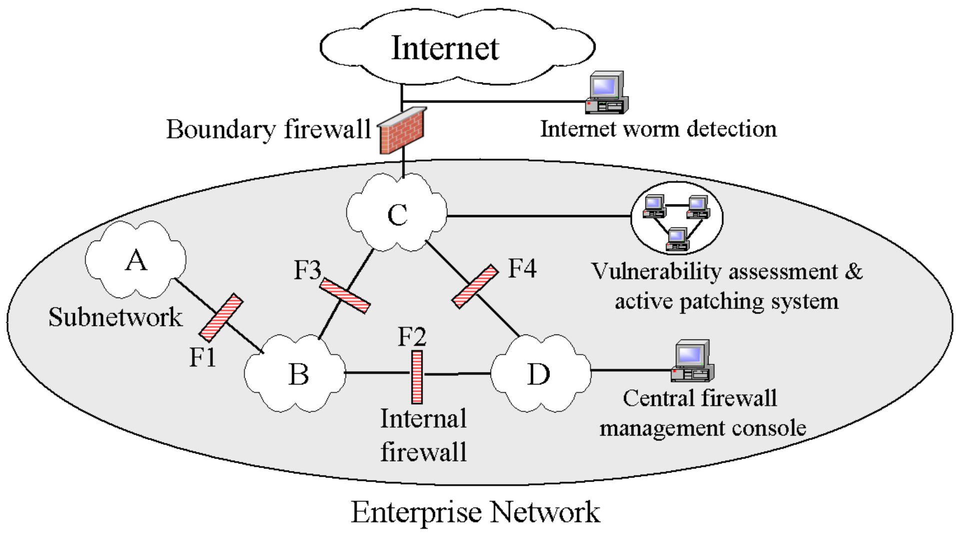

Fig. 1. Architecture of the Firewall Network System for worm defense in an enterprise network

not answer the important issue of how to reduce the complexity of firewall management.

A close work to ours is the "CounterMalice" presented by Staniford [22] for worm defense in enterprise networks. The basic idea is to use CounterMalice boxes to separate an enterprise network into many isolated subnetworks. In a normal situation, these CounterMalice boxes, unlike internal firewalls in our proposed system, impose no constraint on network traffic. They will quarantine an internal infected computer and prevent it from spreading out to other subnetworks once they detect this infected computer.

Nojiri *et al.* [23] presented a "cooperative response" worm defense model in which compromised sites warn "friends" of the presence of a worm, resulting in the friends blocking the worm. Williamson [24] studied worm containment by constraining the outgoing scan rate from infected hosts, which has the effect of decreasing the worm's propagation speed dramatically. To be effective, however, both approaches require global implementation and thus are not suitable for worm defense in an enterprise network.

For early worm detection, Moore *et al.* [5] presented the concept of "network telescope" by using a small fraction of IP space to observe security incidents on the global Internet. Based on a similar monitoring system, Zou *et al.* [26] presented a non-threshold based worm detection method for detecting the exponential growth trend of a worm's propagation. For automatic mitigation of worm attacks, Zou *et al.* [28] presented a feedback dynamic quarantine system that borrows two principles used in epidemic disease control: preemptive quarantine and feedback adjustment. David Nicol [25] studied how various "good" worms could help in worm defense.

III. FIREWALL NETWORK SYSTEM ARCHITECTURE

Fig. 1 illustrates the generic architecture of the Firewall Network System. The Firewall Network System includes several components: (1). Boundary firewall; (2). Internal firewalls; (3). Vulnerability assessment and active patching system; (4). Central firewall management console; and (5). Worm detection systems.

"Boundary firewall" is the traditional firewall currently used in most enterprise networks. It constrains access from the outside Internet to the internal enterprise network. The Firewall Network System does not make any change to the configuration of the boundary firewall. Inside an enterprise network, *"internal firewalls"* divide the network into many isolated subnetworks. Fig. 1 shows the case where an enterprise network is partitioned into four separated subnetworks referred to as "Subnet A", "Subnet B", etc. These internal firewalls are denoted as "F1", "F2", etc.

A vulnerability assessment system is an indispensable security defense to an enterprise [12]. It helps administrators of an enterprise network identify and eliminate vulnerable computers in the network as early as possible. Besides warning users of the vulnerabilities in their computers, administrators can forcedly compromise and patch vulnerable hosts in an enterprise network, especially when a worm is spreading rapidly in the Internet. For these two reasons, the Firewall Network System contains a *"vulnerability assessment and active patching system"*.

For the management of all internal firewalls, the Firewall Network System has a *"central firewall man-*

agement console". Through this console, administrators of an enterprise can easily update firewall rules in all internal firewalls, send command to internal firewalls to quarantine individual host or subnetworks, or collect monitored data from internal firewalls.

To defend against a worm, it must be detected as quickly as possible. Two worm detection systems exist in an enterprise network: the "*Internet worm detection system*" detects the propagation of worms in the global Internet; the "*internal worm detection system*" detects infected hosts within an enterprise network. The "Internet worm detection system" detects a worm based on monitored incoming traffic from the outside Internet to an enterprise network. It can use the monitoring methods presented in [5][26] to detect a worm (by using, for example, the early detection method presented in [26]). The "internal worm detection system" is on the "central firewall management console". It detects internal infected hosts based on monitored data from all internal firewalls.

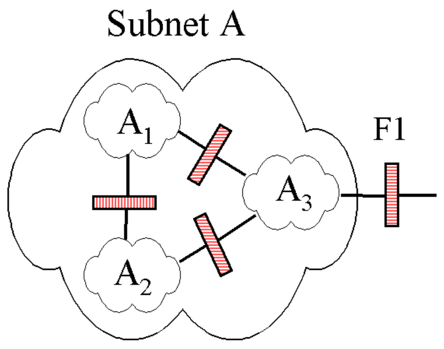

Fig. 2. Low-level Firewall Network System for the Subnet A shown in Fig. 1 ("F1" in this figure is Firewall F1 shown in Fig. 1)

For a large enterprise consisting of tens of thousands of computers around the world, we can implement a "two-level hierarchical Firewall Network System". The high-level Firewall Network System partitions the enterprise network into many isolated but still large subnetworks. For example, each subnetwork could correspond to a branch of the enterprise in one country. Within each large subnetwork, the local administrators of this subnetwork implement a low-level Firewall Network System to further partition this subnetwork into many smaller subnetworks. For example, Fig. 1 shows the high-level Firewall Network System of a large enterprise network and Fig. 2 shows the low-level Firewall Network System for Subnet A ("F1" in this figure is Firewall F1 shown in Fig. 1). The administrators of the enterprise do not need to pay attention to the network traffic within Subnet A — such traffic can be controlled by local administrators. In this way, if a server in Subnetwork A_1 shown in Fig. 2 only provides service to Subnetworks A_1 and A_2, it is treated as a *server* in the low-level Firewall Network System, but as a *client* in the high-level Firewall Network System.

IV. System Design for the Firewall Network System

A. Explicit client/server networking mode

We require that computers in an enterprise network work in the traditional client/server networking mode: every computer is either a client or a server. A "client" means that this computer can send out service requests such as surf the web or log onto a database server, but cannot provide any service to other computers. On the other hand, a "server" can provide specific services to other computers (of course, a server can behave as a client when it sends service requests to other servers).

The advantage of the "client/server" networking mode is that only servers provide services and thus no computer is expected to send a service request to any client computer. After clients and servers are clearly defined, we can implement the access policy in all internal firewalls to simply block all client-to-client and server-to-client *service requests* without affecting the normal usage of the network. Any infected computer in one subnetwork will not be able to infect any other client computer in other subnetworks. Furthermore, the Firewall Network System can very easily detect an infected computer: a service request sent out by a computer is judged abnormal and blocked immediately if it is sent to an IP address (within the enterprise network) other than those predefined servers.

The constraint imposed by the client/server networking mode is that employees of an enterprise cannot arbitrarily set up servers without notifying administrators (if a server is used only within its subnetwork, then employees can set up such a server whenever they want). For an open environment organization such as a university or an ISP, this constraint may not be acceptable. For such organizations, the Firewall Network System is not a suitable solution. Many other enterprises, however, have such a company policy that employees cannot set up servers arbitrarily. For these enterprises, this constraint will not impose much of a burden.

Currently, many enterprises introduce additional firewalls to protect their important servers. With the "client/server" networking mode, the Firewall Network System places more emphasis on protecting client computers — servers are inherently more vulnerable than clients to worm attacks because servers have to accept service requests. For this reason, the current firewalls for protecting servers are still useful when an enterprise implements the Firewall Network System.

Peer-to-peer networking violates the client/server networking mode and hence needs special consideration. We will discuss peer-to-peer system in detail later in Section IV.

B. Explicit firewall access rules with default dropping policy

Internal firewalls restrict network traffic to internal computers and IP addresses. We refer to service requests targeting internal IP addresses of an enterprise network as "*internal service requests*".

The default policy of internal firewalls is to drop any undefined internal service request. We explicitly define firewall rules to allow computers in the predefined subnetworks to access the predefined internal servers. For example, in the network shown in Fig. 1, suppose a new SQL database server is set up in Subnet A to provide database service to all client computers in Subnets A and B. Administrators of the enterprise network then place one firewall rule in Firewall F1 (or all internal firewalls) to allow clients in Subnets A and B to send and only send database service requests to the server. With explicit firewall rules and a default dropping policy, no computer in Subnets C and D is allowed to send service requests to this database server.

With such a firewall access policy, if a client or a server is somehow infected by a worm that needs to compromise a vulnerable host through port x, then the infected host can only possibly infect all vulnerable computers within the same subnetwork and all vulnerable *connectable* servers (servers that provide the infected host the corresponding service on port x) in the enterprise network.

C. Demilitarized Zone (DMZ) for servers

A flaw in the above design arises if we do not separate servers from clients within subnetworks. For example, if each subnetwork contains a vulnerable server and all vulnerable servers in the enterprise network are infected, then each infected server can still infect computers within its own subnetwork. Through this way, the worm can still infect all vulnerable computers within an enterprise network.

For this reason, we separate servers from clients in each subnetwork by internal firewalls. We place all servers of each subnetwork into one or several Demilitarized Zone (DMZ) and connect each zone directly to a nearby internal firewall. One subnetwork could have several DMZs; since a subnetwork could be separated from others by several internal firewalls, each DMZ could connect to its nearest internal firewall for deployment convenience. Because many enterprise networks have a small number of servers compared to the overall number of computers, such an approach is feasible and does not add too much cost.

With such an approach, an infected server will not be able to infect any client computer, even within its own subnetwork. The infected server can only possibly infect vulnerable servers that accept the service requests on the vulnerable port from this server.

D. Same configuration for all internal firewalls

The Firewall Network System for a large enterprise network may include hundreds of internal firewalls. If different internal firewalls have different firewall rules, the firewall configuration will have to consider the positions of internal firewalls, the distribution of servers, and network topology. These considerations will greatly complicate system management and make it not scalable.

This problem is solved by requiring all internal firewalls to have the same configuration — every internal firewall has the same set of firewall rules. With this design, the task of the central firewall management console is very simple: to receive the input of firewall rules from administrators and then duplicate these firewall rules in every internal firewall. Increasing the number of internal firewalls does not add any more complexity to the central firewall management console. This makes the Firewall Network System scalable to large enterprise networks.

Based on the complexity of firewall configuration, we design three types of Firewall Network Systems. From the simplest configuration to the most complicated one, these three types of Firewall Network Systems are:

"*Type-1 Firewall Network System*": For internal service requests, only one firewall rule is used in every internal firewall. We collect the IP addresses (or domain names) of all servers in an enterprise network and place them on one server list. This firewall rule is: allow any computer in the enterprise network to send any internal service request to any computer on the server list. This configuration simply classifies the network into one client group and one server group.

"*Type-2 Firewall Network System*": For internal service requests, one firewall rule is used for one class of servers. Thus if servers in an enterprise network provide n types of services, n firewall rules are used in every internal firewall. For the i-th type of service

($i = 1, 2, \cdots, n$), the corresponding firewall rule is: allow any computer in the enterprise network to send and only send the i-th type of service request to any server providing this type of service. For example, Windows SQL database servers provide database service by accepting service requests to them on TCP/UDP port 1433 and 1434. For this type of service, one firewall rule is used in every internal firewall to allow any computer in the enterprise network to send the database service requests (with destination port as TCP/UDP port 1433 or 1434) to any internal Windows SQL database server defined in the server list.

"*Type-3 Firewall Network System*": For internal service requests, one firewall rule is used for each server. Thus if an enterprise network has n servers, n firewall rules are deployed at every internal firewall. The firewall rule for each server is: allow any computer in the predefined subnetworks that are allowed to use this server to send and only send the corresponding service requests to this server. For example, for the enterprise network shown in Fig. 1, suppose a Web server is set up in Subnet A providing web service to Subnets A and D. Then we place a firewall rule on all internal firewalls (F1 to F4) to allow HTTP service requests on TCP port 80 and 443 passing to this Web server from any IP addresses within Subnets A and D — any computer in Subnets B or C cannot send HTTP service requests to the Web server. In this way, if one computer in Subnets B or C is infected by Code Red, the worm cannot infect this Web server.

E. Other special-purpose firewall access rules

The firewall access policies described above are concerned with internal service requests to internal servers. In this section, we introduce three additional firewall rules that should be placed on every internal firewall.

First, we do not add restrictions to any service request initiated from internal computers to the outside Internet — such a restriction would greatly add complexity to firewall rules and does not help in protecting computers inside an enterprise network (although it might help in protecting the global Internet community). Therefore, in every internal firewall we add the firewall rule: allow any internal computer of an enterprise to send any service request to the outside Internet.

Second, the central firewall management console should be able to connect to all internal firewalls. Therefore, in every internal firewall we add the firewall rule: allow the central firewall management console to send firewall management service requests to any internal firewall in the Firewall Network System.

Third, computers in the vulnerability assessment and active patching system should be able to send any service request to any computer in the enterprise network. Therefore, in every internal firewall we add the firewall rule: allow computers in the vulnerability assessment and active patching system to send any service request to any IP address in an enterprise network.

Considering the above firewall rule, we notice that a UDP-based worm, such as the SQL Slammer, can easily spoof its source IP address. When a UDP-based worm infects an internal computer in an enterprise network, if this computer happens to use an IP address belonging to one of the computers within the vulnerability assessment system as its source IP, then its scans would be able to pass through internal firewalls. Such an event has a very small chance of occurrence because the several IP addresses of the vulnerability assessment system will be kept secret and it is difficult to spoof them blindly. However, this security hole in the Firewall Network System can be avoided by placing one UDP scanning computer in each subnetwork for the purpose of scanning its subnetwork for possible UDP vulnerabilities — these UDP scanning computers communicate with the vulnerability assessment system through TCP communication. In this way, the correct firewall rule for the vulnerability assessment and active patching system is: allow computers in the vulnerability assessment and active patching system to send any *TCP* service request to any IP address in an enterprise network.

For the same reason, the firewall rules in internal firewalls should not rely on source IP addresses for UDP service requests. Thus "type-3 Firewall Network System" discussed previously is only suitable for TCP but not UDP service requests. Fortunately, "type-1" and "type-2" Firewall Network System are suitable for both TCP and UDP service requests because they do not use source IP address in their firewall rules.

Finally, the Firewall Network System can help in defense against many mass-mailing email viruses. The email protocol, Simple Mail Transfer Protocol (SMTP), is used for email exchange between email servers. Users read their emails on their personal computers through email agent software, such as Netscape or Outlook, which uses POP or IMAP protocol to retrieve emails from email servers. Therefore, only email servers are supposed to send out SMTP service requests. However, most mass-mailing email viruses, such as recent SoBig [19] and MyDoom [20], directly send out virus emails from compromised computers based on their own SMTP engines. Based on this observation, the Firewall Network System can place the following firewall rule in every internal firewall: only allow predefined email servers to send out SMTP service requests. In this way, for those email viruses that use their own SMTP engines, an infected computer in an enterprise network cannot send any virus email to any email server outside its subnetwork. The enterprise can not only prevent virus emails from sending out to the global Internet from its network, but also prevent virus emails circulating within the enterprise network — internal virus emails are more dangerous than outside ones because users are more likely to trust emails from their colleagues.

F. Peer-to-peer networking

Peer-to-peer (P2P) networking has become a popular research topic in recent years. In a P2P system, every computer acts as both a server and a client. If an enterprise network deploys P2P systems that contain a large number of computers, the server list in every internal firewall will become large and need to be updated frequently as computers join and leave P2P systems. For this reason, we recommend against using peer-to-peer networking in an enterprise network. (A P2P system can still be used in an enterprise network deployed with the Firewall Network System.)

Since the Firewall Network System does not control traffic within each subnetwork, users can freely set up a P2P system for local usage within a subnetwork. In this way, from the perspective of the Firewall Network System, all computers in the P2P system are treated as clients and need not to be considered in the system design. For example, file sharing in Windows computers is a simple peer-to-peer application. We should restrict the computers that share files with each other to be in one subnetwork — if users want to share files across the boundary of a subnetwork, they should use a Network File Server.

Most P2P systems can be transformed into client/server systems by using one or several *relay* servers (at the cost of speed and storage). For example, if users in different subnetworks want to set up a video conference, the enterprise could provide a video conference relay server — users' client computers directly connect to the relay server without setting up a peer relationship.

If an enterprise heavily relies on P2P systems and cannot transform them to client/server systems, then the Firewall Network System is not suitable for such enterprises.

V. Worm Defense by the Firewall Network System

A. Worm detection

The "Internet worm detection system" detects the presence of a worm in the global Internet by monitoring incoming worm scan traffic to an enterprise network. It can use the monitoring methods presented in [5][26] to detect a worm (by using, for example, the early detection method presented in [26]).

Once the Firewall Network System is set up, the "internal worm detection system" can easily detect an infected host within an enterprise network. Since we have explicitly defined what service requests are allowed in an enterprise network, any worm scan sent from an infected host targeting an IP address in other subnetworks will trigger an alarm so long as the scan does not happen to target a server that allows such a service request from this infected host. In addition, traditional detection methods can still be used to detect the presence of a worm by checking worm scan traffic to the outside Internet.

Staniford [22] introduced a similar system, called *"CounterMalice"*. Compared with the Firewall Network System presented here, CounterMalice has greater difficulty detecting an internal infected host: prior to the detection of an internal infected host, worm scans sent from this infected host to other IP addresses within the enterprise network cannot easily be identified as abnormal since such traffic is allowed by CounterMalice. More importantly, before an infected host in an enterprise network is detected and then blocked by the CounterMalice device, worm scans from this host could possibly have reached vulnerable hosts in other subnetworks and caused infections.

If the Firewall Network System encounters a single abnormal internal service request from a computer, it does not mean that this computer is infected by a worm. This may occur accidentally say if a user installs a server or peer-to-peer software on the computer, or if a user tries to connect to a server that does not provide services to the computer. Therefore, the system should collect a number of abnormal service requests from a host prior to classifying it as infected. The major advantage of the Firewall Network System is that it stops all worm infection attempts sent out from an infected computer to the enterprise network prior to the detection of this infected computer (except infection attempts to internal servers that accept the corresponding service request from the source).

B. Vulnerability assessment and active patching system

As mentioned previously, an enterprise requires a vulnerability assessment system [12] to help administrators identifying vulnerable computers in the network. However, users may still not install patches even after administrators warn them. To prevent serious damage caused by a worm, administrators may want to take aggressive actions, such as actively patching vulnerable computers within their enterprise network. Active patching may have some negative effects such as interrupting operations on vulnerable computers. However, it is a necessary countermeasure so long as its cost is less than the damage caused by a worm.

The active patching system consists of several computers to scan the enterprise network and several servers to provide patches. The scanning computers scan and compromise vulnerable hosts in an enterprise network with exploiting code programmed by security staffs. After compromising a vulnerable host, the exploiting code issues a command to download and install the patch from the patch servers in the patching system. The scanning computers in the active patching system can coordinate to scan the entire IP space of an enterprise network without wasting scanning resources.

One might expect the scanning process to take a long time. This is not the case. For example, if an enterprise has two Class B network space (2^{17} IP addresses) [22], five scanning computers in its active patching system and each scanning computer has a scan rate 100 scans/second, then the active patching system only takes $t = 2^{17}/(5 \times 100) = 4.37$ *minutes* to complete the scanning task.

The scanning time can be further reduced by ignoring unallocated portions of the address space assigned to an enterprise network. In addition, the vulnerability assessment system knows the IP addresses of most vulnerable hosts in an enterprise network from its vulnerability scans. Therefore, the active patching system can first scan and install patches on these known vulnerable computers before scan the remaining IP space of the enterprise network. In this way, the active patching system can patch most vulnerable hosts very quickly.

The recent Nachi worm [17] is a “good” patching worm that attempts to remove Blaster from infected hosts and install patches on them. Within an enterprise network, when many vulnerable hosts are patched and “infected” by a patching worm, the patching worm will consume considerable resources on these patched hosts and generate a large amount of worm traffic. Therefore, we believe that “patching worm” is a bad defense idea, even for an enterprise that has the right to deploy such a worm in its own network.

C. Feedback defense based on security alert level

To defend against worm attacks, Zou *et al.* [28] presented a feedback dynamic quarantine framework that borrows two principles from epidemic disease control: “preemptive quarantine” and “feedback adjustment”. However, it did not discuss how to quantitatively design the optimal feedback control theme. If the feedback control system uses continuous state, to design the feedback system we have to know the accurate dynamic model of the worm propagation system and the quarantine cost function — both the model and the cost function are very hard to derive quantitatively and accurately.

It is more feasible to design a feedback defense system with a finite number of states and a finite number of quarantine control actions. In dealing with terrorism after the 9/11 terrorists’ attack, the US Department of Homeland Security introduced a “five-level homeland security advisory system” [2]. This system describes “green”, “blue”, “yellow”, “orange”, and “red” security levels along with corresponding protective measures that should be taken under these levels — such an advisory system is an example of a finite-state, finite-control feedback defense system.

In the Firewall Network System, we borrow the idea of the five-level security advisory system of Homeland Security [2] and the “feedback adjustment” principle [28] to design a feedback defense system with five security alert levels and five corresponding defense actions.

The choice of security alert level of an enterprise network depends on several factors. These factors include not only the worm detection results from both worm detection systems, but also the answers to the following questions: how serious is a vulnerability? How easy is it to program the worm code? Do there exist any proof-of-concept codes or real testing codes in the Internet? The answers to these questions provide an understanding of the potential security problem before the worm detection systems actually detect a worm.

In the real world, most critical vulnerabilities are first discovered by security researchers; patches are usually made available weeks or months before a worm appears exploiting the corresponding vulnerability. Therefore, in most cases there are a clear time line and a series of development symptoms before a worm propagates in the Internet. Consider Blaster [7] as an example. [14] and [15] summarize the evolution of Blaster: Microsoft provided the patch and publicized the security vulnerability on July 16th, 2003; several days later people began to discuss it and provided proof-of-concept codes in various mailing lists; On July 25 and 26, several groups published ready-to-run version of exploiting codes; and on July 31, attackers tested their worm codes in several universities. Finally ten days later, Blaster appeared and spread across the Internet on August 11.

From our studies of previous worms, we present one possible design of the five-level feedback defense with the Firewall Network System:

- **Green**: This is the situation when neither worm detection system has detected a worm, and no critical vulnerability has been disclosed recently. In this case, if an internal computer sends out forbidden internal service requests, the Firewall Network System sends out a warning message, such as an email, to the user of this computer. No quarantine is implemented at this security level.
- **Blue**: This is the situation when neither worm detection system has detected a worm, but a critical vulnerability has been disclosed recently (one that affects many computers). In this case, if an internal computer sends out forbidden internal service requests on the vulnerable port, a warning message will be sent to the computer's user stating that this computer will be quarantined after, for example, one week, if it continues to send out such illegal service requests.
- **Yellow**: This is the situation when neither worm detection system has detected a worm, but proof-of-concept code is available for compromising a critical vulnerability and people have observed some testing codes in the Internet. In this case, if an internal computer sends out forbidden internal service requests on the vulnerable port, a warning message will be sent to both this computer's user and its local administrator. The computer will be quarantined after a short time if it continues to send out such illegal service requests. According to the feedback principle explained in [28], the time to quarantine decreases accordingly as the threat from a potential worm becomes imminent.
- **Orange**: This is the situation when the "Internet worm detection system" detects a worm spreading in the global Internet, but no such worm traffic has been detected inside the enterprise network. In this case, if an internal computer sends out forbidden internal service requests on the vulnerable port, this computer will be quarantined immediately. In addition, the active patching system can be activated to patch vulnerable hosts in the subnetwork that contains the quarantined computer (not on the scale of the entire enterprise network) to prevent further infection by the infected host in its own subnetwork.
- **Red**: This is the situation when the "internal worm detection system" has detected a worm. It means that the worm is present within the enterprise network and some internal computers have already been infected. In this case, the active patching system is activated to patch all vulnerable hosts in the entire enterprise network to prevent further infection by the worm. If an internal computer sends out forbidden internal service requests on the vulnerable port, this computer and its entire subnetwork will be quarantined immediately.

The feedback defense actions, especially the quarantine and possible active patching in the "orange" and "red" security alert levels, can be issued automatically by the Firewall Network System. In this case, the Firewall Network System becomes a feedback automatic defense system, which has the capability to defend against fast spreading worms.

VI. Worm Propagation Modeling

Under the defense of the Firewall Network System, if one internal computer in an enterprise network is infected by a worm, the worm cannot spread out to other subnetworks through scanning (except that it can possibly infect vulnerable servers in other subnetworks that accept service request on the vulnerable port from the infected computer). In this section, we model worm propagation in one subnetwork that initially contains one or several initially infected computers.

Suppose an enterprise network has Ω IP addresses and is divided into m subnetworks by internal firewalls. Without loss of generality, assume that the subnetwork under consideration is the first subnetwork, which has Ω_1 IP addresses and N_1 vulnerable hosts before a worm infects one or several hosts in it. Denote $I(t)$ as the number of infectious hosts in the subnetwork at time t, $I(0) = I_0$; $S(t)$ as the number of susceptible hosts in the subnetwork at time t, $S(0) = N_1 - I_0$. An infectious host sends out η scans per unit time targeting the entire enterprise network, among which η_1 scans target its own subnetwork.

We model worm propagation under three different active defenses by the Firewall Network System. The first two defenses use the active patching system on the subnetwork under consideration; the third defense does not use the active patching system, but assumes that individually infected hosts in the subnetwork can be quarantined. For the active patching system defense, denote $Q(t)$ as the number of patched hosts that are immune to the worm in the subnetwork, $Q(0) = 0$. The scanning computers in the active patching system scan κ IP addresses per unit time. For the quarantine defense, denote $R(t)$ as the number of quarantined hosts at time t in the subnetwork, $R(0) = 0$.

Define *infection density* α as the fraction of vulnerable hosts eventually infected by a worm in the subnetwork under consideration. The primary objective of active worm defenses implemented on the subnetwork is to decrease a worm's infection density, which is

$$\alpha = [N_1 - S(\infty)]/N_1 \tag{1}$$

TABLE I
NOTATIONS IN THIS PAPER

Symbol	Definition
Ω	Number of IP addresses in an enterprise network
η	Worm scan rate within an enterprise network
m	Number of subnetworks in an enterprise network
N_1	Number of vulnerable hosts in the subnetwork under consideration
Ω_1	Number of IP addresses in the subnetwork
η_1	Worm scan rate within the subnetwork
$I(t)$	Number of infectious hosts in the subnetwork
$S(t)$	Number of susceptible hosts in the subnetwork
$Q(t)$	Number of patched hosts in the subnetwork
κ	Patching system scan rate within the subnetwork
T	Time to quarantine an infected host after it is infected
T_1	Time to finish scanning the subnetwork by active patching-I system
T_2	Time to finish scan by active patching-II system
C	Number of scans observed before quarantine
$R(t)$	Number of quarantined hosts in the subnetwork
α	Fraction of vulnerable hosts in the subnetwork that are eventually infected by the worm
v	Density of vulnerable hosts in the subnetwork under consideration, $v = N_1/\Omega_1$
p	Fraction of vulnerable hosts whose IP addresses are known to active patching-II system

A. Active patching-I: not knowing IP addresses of vulnerable hosts

In the "orange" and "red" security alert levels, the subnetwork that contains an infected host will be quarantined by internal firewalls and patched by the active patching system (beginning at time $t = 0$). First, we analyze the situation when the active patching system does not know IP addresses of vulnerable hosts and scans the whole IP space of the subnetwork. We refer to such an active patching system as "active patching-I" system.

Because the patching system does not waste scans on already scanned IP addresses, it completes its scan of the subnetwork at time

$$T_1 = \Omega_1/\kappa \tag{2}$$

At any time t ($t < T_1$), "active patching-I" system has scanned κt IP addresses and the remaining $\Omega_1 - \kappa t$ IP addresses have not yet been scanned. In the remaining IP space, the density of vulnerable hosts is $\frac{S(t)}{\Omega_1-\kappa t}$. Thus on average, the number of vulnerable hosts to be patched in a unit time at time t is $\kappa\frac{S(t)}{\Omega_1-\kappa t}$ — this is accurate when the worm randomly infects vulnerable hosts in the subnetwork, or when the patching system randomly scans not-scanned IP space in the subnetwork.

Under the "active patching-I" defense, a worm's propagation in one subnetwork follows (based on the simple epidemic model in [1][21] and the worm model in [27]):

$$\begin{aligned}\frac{dS(t)}{dt} &= -\frac{\eta_1}{\Omega_1}S(t)I(t) - \frac{dQ(t)}{dt}\\ \frac{dQ(t)}{dt} &= \kappa\frac{S(t)}{\Omega_1-\kappa t}\\ N_1 &= Q(t)+S(t)+I(t)\\ & 0 \le t < \Omega_1/\kappa\end{aligned} \tag{3}$$

At time $T_1 = \Omega_1/\kappa$, all initially vulnerable hosts are either infected or patched. Thus $S(t) = 0$, $\forall t \in [\Omega_1/\kappa, \infty)$.

B. Active patching-II: known IP addresses of vulnerable hosts

Because the vulnerability assessment system of an enterprise network frequently scans the network to find vulnerable computers, we analyze the situation where the active patching system knows the IP addresses of all vulnerable hosts within the network. We refer to such an active patching system as "active patching-II" system, which only needs to scan IP addresses of all vulnerable hosts in the subnetwork.

Hence the number of IP addresses to be scanned is N_1 instead of Ω_1. "Active patching-II" system can complete its scan of the subnetwork at time

$$T_2 = N_1/\kappa \tag{4}$$

On the other hand, the worm still has the original scanning space Ω_1 because it does not know the IP addresses of vulnerable hosts.

Under the "active patching-II" defense, a worm's propagation in one subnetwork follows:

$$\begin{aligned}\frac{dS(t)}{dt} &= -\frac{\eta_1}{\Omega_1}S(t)I(t) - \frac{dQ(t)}{dt}\\ \frac{dQ(t)}{dt} &= \kappa\frac{S(t)}{N_1-\kappa t}\\ N_1 &= Q(t)+S(t)+I(t)\\ & 0 \le t < N_1/\kappa\end{aligned} \tag{5}$$

At time $T_2 = N_1/\kappa$, all initially vulnerable hosts are either infected or patched. Thus $S(t) = 0$, $\forall t \in [N_1/\kappa, \infty)$.

During the time interval from the vulnerability scan issued by the vulnerability assessment system to the activation of active patching-II system, some vulnerable hosts may change their IP addresses and some new vulnerable hosts may appear. Therefore, it is more realistic to assume that active patching-II system knows the IP addresses of only a fraction of vulnerable hosts in the subnetwork. Denote p as the fraction of vulnerable hosts that active patching-II system knows ($0 \le p \le 1$). Active patching-II system first scans and patches those known pN_1 vulnerable hosts, then the system continues to scan the remaining $\Omega_1 - pN_1$ IP space in the subnetwork attempting to patch the other $(1-p)N_1$ vulnerable hosts.

Under such an "active patching-II" defense, a worm's propagation in one subnetwork follows:

$$\begin{aligned}
\frac{dS(t)}{dt} &= -\frac{\eta_1}{\Omega_1}S(t)I(t) - \frac{dQ(t)}{dt} \\
\frac{dQ(t)}{dt} &= \begin{cases} \kappa\frac{S(t)}{N_1-\kappa t} & , \quad 0 \le t < \frac{pN_1}{\kappa} \\ \kappa\frac{S(t)}{\Omega_1-\kappa t} & , \quad \frac{pN_1}{\kappa} \le t \end{cases} \\
N_1 &= Q(t) + S(t) + I(t) \\
& \quad 0 \le t < \Omega_1/\kappa
\end{aligned} \tag{6}$$

Note that (6) reduces to (3) when $p = 0$ and (5) when $p = 1$.

C. *Worm propagation model based on individual quarantine*

The active patching system can patch vulnerable hosts before they are infected by a worm. However, in most cases remotely patching a computer without notifying the computer's user can interfere with operations on this computer, which will introduce a cost to an enterprise. Not all enterprises can afford the cost of such an aggressive active patching system for worm defense. In addition, in order to remotely patch vulnerable hosts, security staffs in an enterprise need to first program the exploiting code by themselves and test the code carefully before using it in the active patching system, which requires an enterprise to have experienced security staffs.

In the future, an enterprise may have the hardware and software support to enable the quarantine of each infected host. For example, ethernet hubs can be replaced by ethernet switches that can receive command to shut down individual network interface; wireless access devices can be upgraded to be able to receive command to cut off individual connected client.

Now we analyze a defense where the Firewall Network System can quarantine individual infected hosts. Suppose the Firewall Network System will quarantine a suspicious host when internal firewalls have received C illegitimate internal service requests from it. Hence an infected host is quarantined after it sends C scans to other subnetworks in the enterprise network. Denote T as the time for an infected host to be quarantined after it is infected. From the definition of η and η_1, we have:

$$T = \frac{C}{\eta - \eta_1} \tag{7}$$

Under such a quarantine defense, the worm propagation in one subnetwork follows:

$$\begin{aligned}
\frac{dI(t)}{dt} &= \frac{\eta_1}{\Omega_1}S(t)I(t) - \frac{dR(t)}{dt} \\
\frac{dR(t)}{dt} &= \frac{\eta_1}{\Omega_1}S(t-T)I(t-T) \\
N_1 &= R(t) + S(t) + I(t)
\end{aligned} \tag{8}$$

where $dR(t)/dt = 0, t \in [0, T)$. Since all those initially infected hosts $I(0) = I_0$ will be quarantined at time T, $R(T) = I_0$ and $I(T^+) = I(T^-) - I_0$.

VII. Worm Defense Simulation Studies

In this section, we study the performance of the three worm defenses discussed above. Given the parameters in the model (3), (5), (6), and (8), we use Matlab Simulink [18] to derive the numerical solutions of these models.

A. *Active patching defense system*

Suppose an enterprise has been allocated $\Omega = 2^{17}$ IP addresses as used in [22]. A worm's scan rate η_1 within a subnetwork is expected to be small. For example, the uniform-scan worm, Slammer, has 4000/second scan rate [4]. If the subnetwork we consider has 2^{12} IP addresses, then the scan rate of a Slammer infected host targeting within its own subnetwork is only $\eta_1 = 4000/2^{32-12} = 0.0038$/second. Of course, if the worm conducts a local preference scan like the Code Red II and Nimda [22], its scan rate η_1 within its own subnetwork will be much larger.

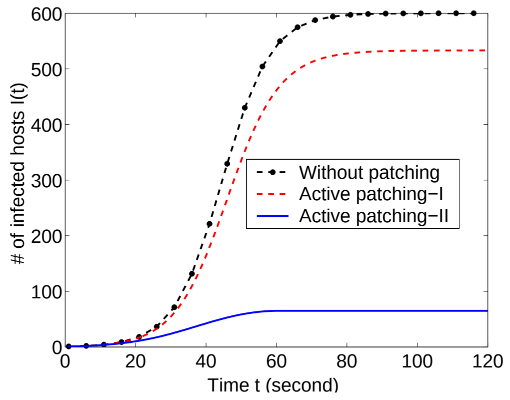

Fig. 3. Worm propagation comparison under the defense of the active patching system ($\Omega_1 = 2^{12}, N_1 = 600, \eta_1 = 1$/sec, $\kappa = 10$/sec, $I_0 = 1$)

In our first experiment, we assume that the enterprise network is equally divided into $m = 32$ subnetworks, i.e., $\Omega_1 = \Omega/m = 4096$; the other parameters are $\eta_1 = 1$/second, $\kappa = 10$/second, and $N_1 = 600$. The scan rate κ is larger than η_1 because those scanning computers in the active patching system use all their scanning power on this subnetwork that has infected hosts; on the other hand, an infected host only uses a small part of its scanning power in this subnetwork — a worm does not know what IP addresses are contained in one subnetwork because the subnetworks are defined logically by internal firewalls, not by IP prefixes.

Fig. 3 shows the worm propagation under the defense of the “active patching-I” system and the “active patching-II” system (5), respectively. In this experiment and the following ones, active patching-II system always means the system described by the model (5) if not mentioned explicitly. We also show in Fig. 3 the original worm propagation without any defense.

In this experiment, because the IP space of the subnetwork is large and active patching-I system requires time $T_1 = \Omega_1/\kappa = 410$ seconds to finish scanning the whole subnetwork, it does not perform well. On the other hand, active patching-II system finishes the patching job by the time $T_2 = N_1/\kappa = 60$ seconds, and hence patches most vulnerable hosts before the worm infects them.

Denote $v = N_1/\Omega_1$ as the density of vulnerable hosts in the subnetwork we consider. In the experiment shown in Fig. 3, the density is $v = 0.15$. The active patching-II system will be more effective when the vulnerable hosts density v decreases because the time for the system to finish patching is $T_2 = v\Omega_1/\kappa$. In order to study the effect of the vulnerable hosts density v, we vary the value of v from 0.04 to 0.5 in steps of 0.02 in our next experiment where the other parameters are the same as used in Fig. 3.

The experiment results are shown in Fig. 4. This figure shows that active patching-II system works best when the density of vulnerable hosts v is low. Fortunately, because an enterprise usually only uses a part of its allocated IP space and because not all computers are vulnerable to a particular worm, in practice the density v is usually very small [22].

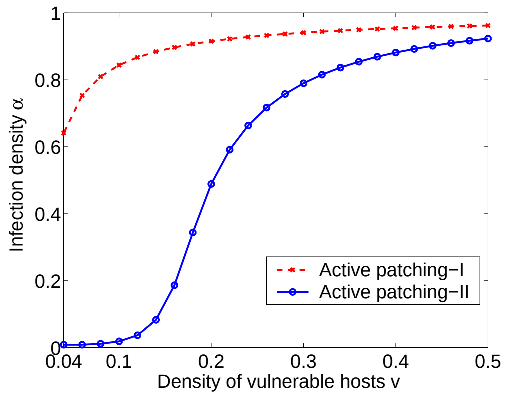

Fig. 4. Infection density α under different density of vulnerable hosts v ($\Omega_1 = 2^{12}, \eta_1 = 1$/sec, $\kappa = 10$/sec, $I_0 = 1$, $N_1 = v\Omega_1$)

Until now we have studied the active patching systems described by model (3) and (5). The more realistic patching system is described by model (6) where we know the IP addresses of part of vulnerable hosts. Intuitively, when we know more vulnerable hosts' IP addresses (i.e., increasing p), we can patch faster and prevent more vulnerable hosts from being infected. As we increase p from 0 to 1, the worm's infection density α should lie between the two curves in Fig. 4. In Fig. 5, we show the infection density α as a function of the value p under three different densities of vulnerable hosts v. This figure is consistent with Fig. 4: as the density of vulnerable hosts v decreases, it is more effective to know vulnerable hosts' IP addresses. If the density v is as high as 0.4, Fig. 5 shows that there will be no difference whether we know only 20% or the complete 100% of vulnerable hosts' IP addresses.

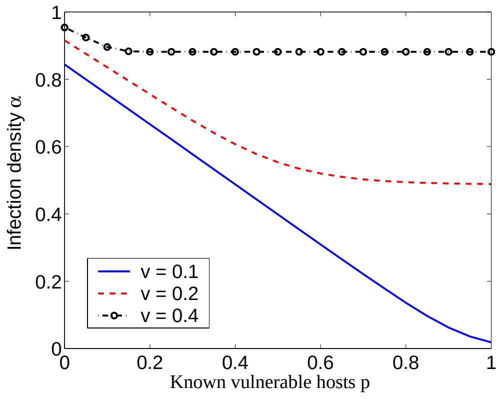

Fig. 5. Infection density α when we know different fraction of vulnerable hosts' IP addresses ($\Omega_1 = 2^{12}, \eta_1 = 1$/sec, $\kappa = 10$/sec, $I_0 = 1$)

One should not interpret our results to mean that active patching-I system is not effective. If scanning computers in the active patching system have a large scan rate and the subnetwork defined by internal firewalls is small, then active patching-I system can finish patching quickly before most vulnerable hosts are infected. Suppose the subnetwork has $\Omega_1 = 1024$ and $N_1 = 500$. We still use $\eta_1 = 1$/second, $I_0 = 1$ but vary κ from 5/sec to 100/sec in steps of 5. The experiment results are shown in Fig. 6. This figure shows that active patching-I system is effective when the scan rate of the active patching system is higher compared with the worm's subnetwork scan rate η_1.

B. *Quarantine defense system*

Now we study the quarantine defense system described by (8). In the subnetwork described in Fig. 4 ($\Omega = 2^{12}, N_1 = 600, I_0 = 1$), suppose an infected host

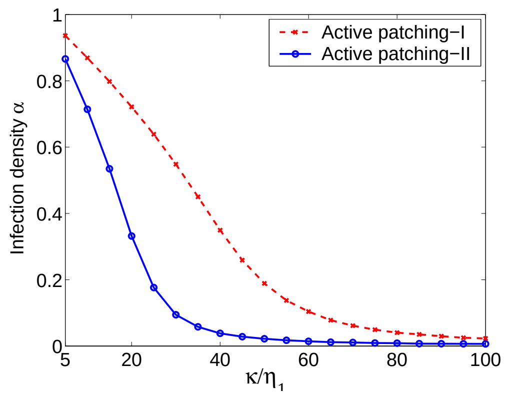

Fig. 6. Infection density α under different patching scan rate κ ($\Omega_1 = 2^{10}, N_1 = 500, \eta_1 = 1$/sec, $I_0 = 1$)

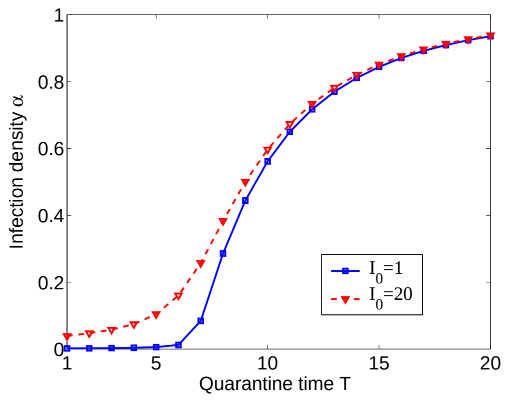

Fig. 8. Worm infection density α under the quarantine defense with different quarantine time T ($\Omega_1 = 2^{12}, N_1 = 600, \eta_1 = 1$/sec)

has a scan rate $\eta = 2$/sec to the entire enterprise network and $\eta_1 = 1$/sec to its own subnetwork; and the Firewall Network System will quarantine an infected host after observing $C = 10$ scans on internal firewalls. According to (7), in this system an infected host will be quarantined after $T = 10$ seconds.

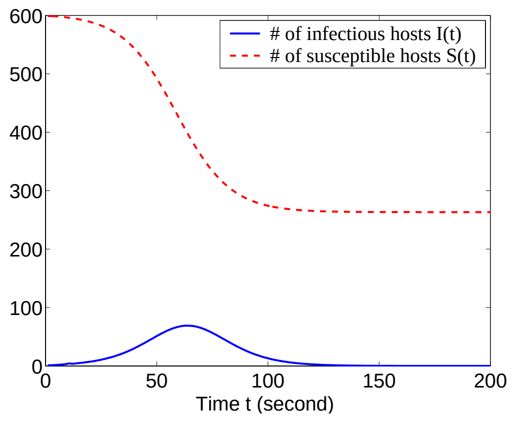

Fig. 7. Worm propagation under the quarantine defense system described by Equation (8) ($\Omega_1 = 2^{12}, N_1 = 600, \eta_1 = 1$/sec, $I_0 = 1, T = 10$/sec)

Fig. 7 shows the worm propagation under this quarantine defense system. After reaching a peak at time $t = 64$ seconds, the number of infectious hosts, $I(t)$, drops down gradually to zero. In the end there are $S(\infty) = 263$ uncompromised susceptible hosts, thus the infection density $\alpha = 0.56$.

In reality, the quarantine defense system usually can do much a better job than what shown in Fig. 7. First, since internal firewalls have explicit access rules, we do not need to wait to receive $C = 10$ scans to quarantine an infected host. Second, the subnetworks are defined by internal firewalls, not by IP prefixes. Thus a worm does not know what IP addresses are within its own subnetwork, which makes it difficult for an infected host to send most of its scans within its own subnetwork. For these two reasons, the time to quarantine T usually is very short. For example, suppose the worm in the experiment shown in Fig. 7 uniformly scans the enterprise network, then its time to quarantine is $T = 10/(32-1) = 0.32$ seconds ($m = 32$ thus $\eta = 32$/sec). To study how T affects the quarantine defense system, we conduct an experiment with the same parameters used in Fig. 7 ($\Omega_1 = 2^{12}, N_1 = 600, \eta_1 = 1/$sec) but varying T (we also consider two situations where the number of initially infected hosts is $I_0 = 1$ and $I_0 = 20$, respectively). The worm's infection density v under different T is shown in Fig. 8. This figure shows that the quarantine defense system works best when the time needed to quarantine infectious hosts is short compared with the worm's subnetwork scan rate η_1.

VIII. Conclusions

In this paper, we present a "Firewall Network System" for worm defense in enterprise networks. The Firewall Network System uses firewalls to divide an enterprise network into many isolated subnetworks. All computers in an enterprise network are classified as either clients or servers: all service requests sent to internal IP addresses of an enterprise network will be blocked by internal firewalls if they target non-server computers, or servers that do not provide the corresponding service. In this way, the Firewall Network System removes most worm infection paths in an enterprise network, making worm detection much easier.

In our design, all internal firewalls in the Firewall Network System are designed to have the same set of firewall rules, which means the Firewall Network System is scalable to large enterprise networks and easily managed by a central firewall management console. We also propose a five-level feedback worm defense strategy based on the "feedback quarantine" principle from epidemic disease control [28] and the idea of the five-level US "Homeland Security Advisory System" [2]. Furthermore, we model and analyze three defense strategies based on either active patching or quarantine.

Even without any active defense from the Firewall Network System, no matter how fast a worm can propagate (through scanning and direct compromising), it can infect at most all vulnerable hosts in its subnetwork and vulnerable servers in the enterprise network that accept service requests on the vulnerable port from the infected subnetwork.

The Firewall Network System is not an omnipotent worm defense solution for all enterprises. The major constraint of the Firewall Network System is that employees of an enterprise cannot set up servers arbitrarily without notifying administrators. For an open environment organization such as a university or an ISP, this constraint may not be acceptable. Because of the client/server networking requirement, the Firewall Network System is not suitable for enterprises that heavily rely on peer-to-peer networking and cannot transform their P2P systems to client/server systems.

However, many enterprises do not depend on peer-to-peer networking and have the company policy that employees cannot set up servers arbitrarily. For these enterprises, the Firewall Network System is suitable and worth to be deployed considering its security benefit. Of course, many detailed technical issues need to be discussed before the Firewall Network System is deployed in practice.

IX. Acknowledgement

This work is supported in part by ARO contract DAAD19-01-1-061, the National Science Foundation Grants ANI9980552, and by DARPA contract F30602-00-0554.

References

[1] D. J. Daley and J. Gani. *Epidemic Modelling: An Introduction*. Cambridge University Press, 1999.

[2] US Homeland Security Advisory System. http://www.dhs.gov/dhspublic/display?theme=29

[3] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham. A Taxonomy of Computer Worms. In *ACM CCS Workshop on Rapid Mal-code (WORM'03)*, Oct. 27, 2003.

[4] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the Slammer Worm. *IEEE Security and Privacy Magazine*, 1(4):33-39, July 2003.

[5] D. Moore. Network Telescopes: Observing Small or Distant Security Events. In *11th USENIX Security Symposium*, 2002.

[6] CAIDA. Dynamic Graphs of the Nimda worm. http://www.caida.org/dynamic/analysis/security/nimda/

[7] eEye Digital Security. Blaster Worm Analysis. http://www.eeye.com/html/Research/Advisories/AL20030811.html

[8] eEye Digital Security. .ida "Code Red" Worm. 2001. http://www.eeye.com/html/Research/Advisories/AL20010717.html

[9] J. Snyder. Roadblocks to Defense-in-Depth. In *Information Security Magazine*, June, 2003.

[10] J. Snyder. Turning the Network Inside Out. In *Information Security Magazine*, June, 2003.

[11] D. Houser. Network Security: Submarine Warfare. In *Information Security Magazine*, August, 2003.

[12] M. Andress. Buyer's Guide: Vulnerability-assessment tools. http://nwfusion.com/reviews/2002/0204bgtoc.html

[13] NSA Security Recommendation Guides. Defense in Depth: A practical strategy for achieving Information Assurance in todays highly networked environments. http://www.nsa.gov/snac/support/guides/sd-1.pdf

[14] 2003 Evolution of DCOM-RPC Exploit. http://www.sbslinks.com/timeline.htm

[15] Farm9.com: Timeline to CyberCrime. http://farm9.com/pdf/CyberCrime_Timeline.pdf

[16] Microsoft Security Bulletin MS03-026, July 16, 2003. http://microsoft.com/security/security_bulletins/ms03-026.asp

[17] Microsoft. What You Should Know About the Nachi Worm. August 18, 2003. http://www.microsoft.com/security/antivirus/nachi.asp

[18] Mathworks: Simulink. http://www.mathworks.com/products/simulink/

[19] CERT Incident Note IN-2003-03: W32/Sobig.F Worm. http://www.cert.org/incident_notes/IN-2003-03.html

[20] CERT Incident Note IN-2004-01: W32/Novarg.A Virus. http://www.cert.org/incident_notes/IN-2004-01.html

[21] 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.

[22] S. Staniford. Containment of Scanning Worms in Enterprise Networks. To appear in *Journal of Computer Security*, 2003.

[23] D. Nojiri, J. Rowe, and K. Levitt. Cooperative Response Strategies for Large Scale Attack Mitigation. In *3rd DARPA Information Survivability Conference and Exhibition*, Apr. 22-24, Washington DC, 2003.

[24] M. M. Williamson. Throttling Viruses: Restricting Propagation to Defeat Mobile Malicious Code. In *18th Annual Computer Security Applications Conference*, Dec. 09-13, San Diego, 2002.

[25] David M. Nicol. Models of Internet Worm Defense. In *IMA Workshop 4: Measurement, Modeling and Analysis of the Internet*, Jan 12-16, 2004.

[26] C. C. Zou, L. Gao, W. Gong, and D. Towsley. Monitoring and Early Warning for Internet Worms. In *10th ACM Conference on Computer and Communication Security (CCS'03)*, Oct. 27-31, Washington DC, 2003.

[27] C. C. Zou, D. Towsley, and W. Gong. On the Performance of Internet Worm Scanning Strategies. *Univ. Massachusetts Amherst Technical Report: TR-03-CSE-07*, November, 2003.

[28] C. C. Zou, W. Gong, and D. Towsley. Worm Propagation Modeling and Analysis under Dynamic Quarantine Defense. In *ACM CCS Workshop on Rapid Malcode (WORM'03)*, Oct. 27, Washington DC, 2003.