Missouri S&T Library and Learning Resources

Scholars' Mine

Masters Theses

Student Theses and Dissertations

Spring 2007

Design and implementation of a hardened distributed network endpoint security system for improving the security of internet protocol-based networks

William Dee Atkins

Follow this and additional works at: https://scholarsmine.mst.edu/masters_theses

 Part of the Computer Engineering Commons

Department:

Recommended Citation

Atkins, William Dee, "Design and implementation of a hardened distributed network endpoint security system for improving the security of internet protocol-based networks" (2007). *Masters Theses*. 4548.
https://scholarsmine.mst.edu/masters_theses/4548

This thesis is brought to you by Scholars' Mine, a service of the Missouri S&T Library and Learning Resources. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact scholarsmine@mst.edu.

DESIGN AND IMPLEMENTATION OF A HARDENED DISTRIBUTED NETWORK ENDPOINT SECURITY SYSTEM FOR IMPROVING THE SECURITY OF INTERNET PROTOCOL-BASED NETWORKS

by

WILLIAM DEE ATKINS

A THESIS

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI-ROLLA

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN COMPUTER ENGINEERING

2007

Approved by

_____	_____
Ann K. Miller, Advisor	R. Joe Stanley

_____

Daniel R. Tauritz

Copyright 2007
William D. Atkins
All Rights Reserved

ABSTRACT

The majority of modern computer networks utilize Ethernet-based hardware and rely on the Internet Protocol for the routing of network traffic. Recent trends show that many of these networks are subsequently connected to the Internet so as to enable remote access and administration of the devices connected to such networks. Because connection of these networks to the Internet renders them open to attack by malicious third parties, firewalls—devices that filter traffic flowing between networks according to administrator-defined policies—are commonly used as a means of defending such networks. While mostly effective at preventing attacks from malicious outsiders via the Internet, these firewalls, by virtue of their placement at the border between internal networks and the Internet, provide no capabilities to protect network-connected systems and devices against malicious insiders or third parties that have in some way commandeered control over even a single internally-connected system or device.

This thesis proposes a distributed approach to securing computer networks by delegating the role of a conventional firewall to a collection of nodes and controllers placed throughout the networks they are intended to protect from attack. This distributed firewall system is a specific application of a generalized distributed system framework that is also proposed in this thesis. The design and implementation of both the generalized framework and the application of the framework in creating a distributed firewall system for use on Ethernet-based networks that rely on the Internet Protocol are discussed. Conclusions based upon the preliminary implementation of the proposed systems are given along with future directions.

ACKNOWLEDGMENTS

I wish to sincerely thank Dr. Ann Miller, the Cynthia Tang Missouri Distinguished Professor of Electrical and Computer Engineering at the University of Missouri–Rolla, for her dedication and support as my advisor through my graduate tenure, and for her funding of this research via a graduate research assistantship in the Trustworthy Systems Laboratory. Certainly without her leadership, advice, and support this thesis and the research presented therein would not have been possible.

I also thank Dr. R. Joe Stanley, Associate Professor of Computer Engineering, and Dr. Daniel R. Tauritz, Assistant Professor of Computer Science, both of the University of Missouri–Rolla, for their voluntary participation as members of my thesis committee. Their insights and advice proved invaluable in making this publication successful.

To my parents, Dennis and Rhonda, and my brother, Greg: You have always believed in me, and it has been your constant encouragement that has driven me to achieve my very best. My most heartfelt appreciation for everything that you have done for me since my very beginning. I love you all.

For their assistance and support, I also wish to acknowledge and thank Robert L. Hutchinson, Joseph H. Maestas, Shannon L. Ward, Daniel W. Loughridge, Gregory J. Atkins, and Rolando G. Melgoza.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1. INTRODUCTION

1.1. BACKGROUND

1.1.1. The Role of Modern Networked Systems. Perhaps one of the most momentous events in the history of computer networking was the creation of a de facto standard termed DIX Ethernet, released in 1980 as a joint effort of Digital Equipment Corporation, Intel Corporation, and Xerox. DIX Ethernet was the basis for the IEEE 802.3 Ethernet standard that was approved in 1983. While both the physical media by which Ethernet networks are constructed and the speeds at which the networks are capable of operating has changed since the initial release of the 802.3 specifications, the Ethernet standard has remained wildly popular given its scalability, low cost, and tremendous flexibility in meeting the needs of a broad computing audience. Today, the majority of local area networks (LANs) are Ethernet based, with connections into the network typically operating at a speed of 100Mbps over a dedicated unshielded twisted pair of copper cabling run from a network hub or switch to each end system.

As businesses, government and educational institutions, and other medium- to large-sized organizations implemented networks, the role that computers played within such organizations dramatically expanded. Collaboration between personnel was simplified, and access to the data necessary for operations continuity became nearly instantaneous. Centralized storage of data became possible, allowing for the possibility of data sharing between individuals and subgroups within networked organizations. Operating systems on servers and end systems adapted accordingly to the presence of the network, providing the ability for users to roam between network-connected workstations while maintaining access to their data. Electronic messaging and other services were deployed, further increasing productivity and simultaneously making the availability of the network increasingly important to operations continuity.

By the mid-1990s, a vast number of medium- to large-sized organizations had implemented Ethernet-based networks at their sites. Large organizations typically had linked together LANs at various branch sites via dedicated wide area network (WAN) links. Such WAN connections often ran at much slower speeds than the LANs they connected, but provided even stronger collaboration and productivity increases, as data

objects at any connected branch location could be accessed from any other connected branch location. As connectivity to the Internet became available to such organizations, many opted to use the Transmission Control Protocol/Internet Protocol (TCP/IP) suite for the network layer addressing and routing schemes on their LANs to connect them to the Internet, allowing for a reduction in WAN connection costs while opening to their networked personnel a wealth of information from other organizations. By 2000, the Internet had massively grown in size, with organizations and users from every corner of the world able to share and access unprecedented amounts of data at unimaginable speeds.

Computers and the networks that connect them play a vital role in the lives of everyone today. Service-oriented businesses, such as financial institutions, insurance companies, stock brokerages, and airlines, use computer networks to access data stored in massive centralized databases. Networking allows these businesses to maintain real-time and near-real-time information on customer accounts, service rates, stock market prices and trends, and an endless list of other record categories. A diversity of organizations use computer networks when collaborating across vastly geographically separated sites. Oftentimes networks allow digital systems present at unmanned sites to relay surveillance and status information back to central manned control rooms that can appropriately dispatch security teams to handle breaches and other incidents. Educational institutions rely on the Internet as an enabler for distance learning courses and as a mechanism for linking together international research teams. Virtually every industry, inclusive of electric, gas, oil, water, and sewer utilities and all varieties of manufacturers, relies on a breed of networks known as process control systems—which are also often Ethernet based—to centrally monitor and control physical plant operations from a single operations and control center.

Computer networks, particularly Ethernet and TCP/IP based, are ubiquitous across all facets of modern societies. Many of these networks are massively complex despite their design to be agnostic of the data they send between systems, keeping the "intelligence" at its endpoints. This design has allowed these networks to be tremendously flexible in their application, but has also prevented them from defending

themselves and the devices attached to them against accidental misconfiguration and attack by malicious entities.

1.1.2. The Need to Secure Networked Systems. As organizations began connecting their networks to the Internet, the need to secure them against attack from malicious external entities became apparent. For businesses and service-oriented organizations, the Internet not only connected remote branches, customers, and business partners, it also connected competitors and cyber-savvy criminals that stood to gain a competitive advantage or profit from covertly performing corporate espionage or gaining access to their now Internet-connected systems that contained everything from corporate organization charts to coveted, lucrative trade secrets. Government organizations had to address the possibility of surveillance of their computer and network operations by unfriendly nation-states and antigovernment groups to protect their national security. Manufacturing and utility industries that relied on the Internet as a means of connecting geographically separated sites were required to consider countermeasures to mitigate the possibility of attacks targetting their process control systems, as maliciously triggered events could cause serious damage to their operations and the environment, i.e., an attacker instructing a process control system to overload a high-pressure natural gas pipeline or, as documented as having occurred in Australia, to pump raw sewage into public water supplies [1].

The necessity for security in these interconnected networks erupted mostly from security being a nonissue when Ethernet, the Internet, and the protocols upon which these networks operate were designed. In the early 1980s, when Ethernet and the TCP/IP suite were being developed, the fields of computing and even moreso of computer networking were composed of a specialized and small group of corporations, research institutions, and government agencies. The pioneering days of computing and computer networking had about them an atmosphere of trust and openness; the focus of researchers and developers, therefore, was on creating functionality rather than on adding security mechanisms. As more organizations began connecting their networks to the Internet, however, the atmosphere had shifted; it was as if a community of 1000 people had experienced a 50,000-fold increase in its population in less than a decade.

The first major Internet security event occurred on November 2, 1988, when Robert Morris, Jr. released what is now referred to as the "Morris Worm," a self-replicating computer virus of sorts that exploited vulnerabilities in Internet-connected hosts running specific versions of network services on varients of the 4 BSD UNIX operating system [2]. The worm, though believed not to be intended as malicious, slowed thousands of Internet hosts to a crawl, disrupting normal Internet connectivity and operations for many days. Since the onslaught of the Morris Worm, it was apparent to the members of the Internet community that no longer was that community small enough to be inherently trusted.

Today, computer and network security is a very serious issue. Software vulnerabilities in applications running on Internet-connected computers can allow attackers to commandeer complete control over them. Attackers can launch automated probing tools to rapidly scan millions of networked hosts for vulnerabilities and then use packages developed by third-parties to exploit those vulnerabilities without any detailed or specialized knowledge. Unsecured networks can be surveilled remotely by malicious entities from different continents, and information flowing across that network may be used in identity theft, fraud, and extortion schemes. Owners of systems that have been remotely commandeered and used to launch attacks against other networked systems may find themselves unable to prove their innocence and be held legally liable for damages caused. As the role of networked computer systems and the Internet has grown, so has the necessity for securing those systems which, unfortunately, has proved to be an astonishingly difficult task.

1.1.3. Thesis Outline. Providing effective and sensible means of defending networked systems against attacks such as those described above is the focus of this thesis. Specifically, a generalized distributed security system framework is proposed, and an application of that framework in the form of a distributed firewall system intended to shield end systems present on Ethernet-based networks utilizing the TCP/IP suite is subsequently proposed and analyzed.

The remainder of this thesis is organized as follows. Section 1.2 presents a history on the concept of conventional network firewalls, how such firewalls are commonly used to secure networks against attack, and how conventional firewalls fall short of protecting

networks against a variety of threats from both external and inside malicious entities. The section also introduces the concept of distributed firewalls by providing a history of the concept and later examining how previously proposed and implemented distributed firewall systems also fall short in protecting networks against certain categories of threats.

Section 2 of this thesis addresses the approach and methodology used in the design and implementation of both the generalized distributed system framework and the distributed firewall system proposed. The goals influencing the design of the generalized distributed system framework are presented and discussed in the order of their priority. Details on the hardware platforms and software packages used in the construction of the proposed distributed system framework and the distributed firewall system are given.

Section 3 presents an analysis of the distributed system framework and the distributed firewall system implementation associated with this thesis. A summary of thesis and results is provided, conclusions on the systems are drawn, and future directions for this thesis and its associated research are given.

1.2. DESCRIPTION OF PROBLEM

As organizations connected their LANs to the Internet to permit connectivity between branch locations and to open their personnel to a wealth of knowledge of unprecedented magnitude, it became clear that protecting devices attached to Internet-connected networks was critical in maintaining positive control over those devices and the data they processed. The most prominent solution developed to date to secure such networks from would-be intruders and attackers is the firewall, which both provides LANs a means of interconnecting with other networks and simultaneously segregates networks connected to it from each other. This section characterizes some of the shortcomings in using current firewall technology to secure interconnected networks, in particular networks connected to the Internet.

Throughout the remainder of this thesis, the term ‘conventional firewall’ will refer to a member of the family of traffic filtering devices whose configurations are not managed via a central controller and whose behavior does not correspond with that of

similar devices such that the collection forms a single system. The term ‘distributed firewall’ will refer to a collection of traffic filtering devices whose configurations are managed by a central controller and whose individual components behave such that they collectively form a single, distributed system.

1.2.1. History of the Conventional Network Firewall Concept. After the attack of the Morris Worm on Internet-connected hosts in 1988, momentum behind the concept of the network firewall began. The term ‘firewall’ was likely adopted from its reference to physical barriers intended to prevent the spread of fire damage in structures and automobiles. In terms of computer networking, a firewall is, “[a] component or set of components that restricts access between a protected network and the Internet, or between other sets of networks [3].” In their definitive text, Bellovin and Cheswick assign the following properties to a firewall: it is a single point between two or more networks through which all traffic flowing between those networks must traverse, the device can control and may authenticate traffic flowing through it, and the device may log all such traffic [4]. These three properties provide an excellent basis for firewall devices today, however, it should be noted that additions, deletions, and modifications of these three properties is common. Bellovin later summarized that, "Firewalls are barriers between 'us' and 'them' for arbitrary values of 'them,'" a statement that better characterizes current firewall technology.

The first firewalls were primitive in comparison to those in existence today, and were more akin to routers than their current successors. The first firewalls routed traffic between two or more networks contingent upon the traffic matching one or more IP filtering rules. Generally, these rule sets instructed the firewall to allow traffic from internal, “trusted” networks, to flow outward to external, “untrusted” networks such as the Internet without restriction. However, traffic from untrusted networks was not allowed to flow into trusted networks unless such traffic was in response to an existing connection initiated by a host on the trusted network for which the traffic was destined. This behavior followed Bellovin’s basic criterion of keeping “them” separate from “us” but otherwise provided little flexibility in managing the flow of traffic.

The next generation of firewalls provided improved granularity in traffic management and were built upon bastion hosts, which are “computer system[s] that must

be highly secured because [they are] exposed to the Internet and [are] a main point of contact for users of internal networks." [3] The first commercial firewall of this type was produced by Digital Equipment Corporation (DEC) and was named the DEC Secure External Access Link (SEAL). DEC SEAL also relied on the concept of application gateways, or proxies, that accept traffic attempting to traverse the firewall and, based upon the application that generated the traffic, determine if the traffic is valid and approved such that it should be forwarded to its intended destination. The capability to classify traffic based upon its application and validity gave administrators the ability to more heavily restrict "riskier" traffic flowing between networks while more liberally allowing safer traffic to pass.

Firewall packages—both in software-only and combined hardware/software forms—progressed rapidly throughout the early and mid-1990s. Enhancements intent on simplifying firewall configuration began with the release of the Firewall-1 product by Check Point in 1994. Rather than relying on administrators defining rules and network elements using text-based configuration files, the Firewall-1 package provided a graphical user interface, with icons, color-coding, and mouse support, the purpose of which was to streamline firewall configuration and monitoring. As with the rest of the computer software and hardware markets, the ease of use and capabilities of firewall packages continues flourish, with firewalls demonstrating advanced capabilities in remote administration and notification, redundancy and fail-over, traffic encryption, and adaptive scanning and filtering of content present within the traffic traversing them.

Two primary categories of firewalls exist today. Both categories are variations on the same general theme already presented in that the firewalls are placed at the border of two or more networks that have varying security requirements and, based upon some set of defined rules, determine whether or not to allow traffic in question to flow through to its intended destination. These categories are covered in brief as they are well documented in other literature sources and only a general understanding is necessary for readers of this thesis.

The first category of firewalls is dubbed as packet filtering. Data on networks utilizing the TCP/IP protocol suite is sent encapsulated in units known as packets at the network layer of the OSI reference model (see Appendix A). Packets include a header

portion that provides metadata associated with the actual data payload of the packet. This metadata includes the IP address of the host from which the packet was sent, the IP address of the host to which the packet is destined, the type of the packet (TCP, UDP, or ICMP), the (logical) port from which the packet was sent on the sending device, the (logical) port on which the application running on the receiving device can be reached, and the overall size of the packet in bytes. Packet filtering firewalls examine this header information (and optionally verify its correctness) to determine whether the packet is to be forwarded on to its destination based upon established filtering rules. The packet filtering category can be further divided into two broad subcategories, stateful and stateless packet filtering. Stateless packet filtering implies that static filtering rules are applied to all packets attempting to traverse the firewall, while stateful packet filtering implies static filtering rules are applied to the first packet of a connection and, based upon its acceptance under the static rule set, dynamic rules are applied to subsequent packets associated with the connection [3].

The other category of firewalls, application gateway or proxy-based, has already been briefly discussed. Proxy-based firewalls operate at the application layer (Appendix A), the layer within which computer programs interact with the network. To properly operate, proxy-based firewalls are required to understand how network-connected instances of specific applications communicate. The premise is to examine data being exchanged between instances of an application via the network to determine if their data exchanges are valid and properly formatted. If this is the case, the traffic is typically allowed to pass through the firewall onto its intended destination; otherwise, the traffic is typically dropped and/or logged, depending upon configuration. One obvious limitation of proxy-based firewalls is that the software running on them must include definitions of data exchanges for each application to be allowed through; oftentimes, software version updates may require these definitions to also be updated. Despite their limitations and intrusiveness, proxy-based firewalls are revered as secure given their validation of the data contained within packets entering and leaving trusted networks.

1.2.2. Securing Networks Using Conventional Firewalls. Although the deployment of conventional firewalls may vary widely to accommodate the security needs of organizations and the individual networks attached to them, general

consistencies between deployments do exist. The focus here is on examining these consistencies rather than providing details on specific deployments; however, examples of specific deployments will be given for context and completeness.

Conventional firewalls are placed at the point of intersection between two or more networks. While segregating two or more networks using a conventional firewall may be done for other than security purposes, such as to prevent broadcast traffic on the networks from creating congestion, the deployment of conventional firewalls is most often done for security reasons and therefore in this thesis the focus is on the security services provided by firewalls. The primary role of a conventional firewall is to in some manner filter traffic flowing between networks whose levels of trust and security differ but for some reason require connectivity to each other. In the case of a conventional firewall connecting exactly two networks, naming conventions commonly dub the network with a lower level of trust as "untrusted" or "external," while the network with a higher level of trust is called "trusted" or "internal." Obviously, this naming convention ceases to be clear when more than two networks are connected by a single conventional firewall; however, following Bellovin's description of firewalls as being devices intended to separate 'us' from 'them' for arbitrary values of 'them,' it is often clear from context which networks connected to a firewall require greater protection and which do not. In the case of a firewall connecting one or more networks to the Internet, the Internet is considered to be the untrusted network and the remaining, internal network(s) are considered as trusted but each likely with a different degree of trust.

Following the trend of organizations connecting their LANs to the Internet but prior to the boom of electronic commerce, a typical deployment of conventional firewalls was rather straightforward—simply place a conventional firewall at the point where the internal LAN connects to the Internet. Figure 1.1 illustrates this basic case. In this configuration, all servers, workstations, printers, and other devices are interconnected via the internal LAN but are separated from the Internet via a single conventional firewall.

As more organizations began hosting websites and using the Internet to work with partner organizations, however, the question of where to place servers that provided these public-facing Internet services was raised. Placing such servers between the router and the conventional firewall was risky, as it exposed the servers to unrestricted attack from

the Internet, and placing them on the internal LAN was also risky, as it would require connections to the servers to pass from the Internet into the internal network. To solve this problem, the military concept of the demilitarized zone—an area between military powers that is considered to be neither inside nor outside the borders of either power—was adopted. In terms of computer networking, a demilitarized zone (DMZ) is a network protected by a firewall from untrusted networks but whose connections to and from trusted internal networks is also filtered. In a sense, a DMZ network is considered neither as being internal nor external and therefore is assigned an intermediate level of trust [5]. Two primary methods exist in establishing a DMZ—adding a third network interface to a typical conventional firewall separating a trusted network from an untrusted one, or utilizing two separate conventional firewalls. Figure 1.2 illustrates the case where a single conventional firewall with three interfaces is used to create a DMZ, and Figure 1.3 the case where two conventional firewalls are used.

A DMZ provides a suitable environment for public-facing servers because entities connecting from the internal LAN can be given more liberal access while entities connecting from the Internet can be granted only the minimum amount of access necessary for the services to operate properly. In addition, should a malicious entity gain control over a DMZ-connected system via some attack vector, a properly configured firewall should prevent that entity from using the compromised server to directly attack systems on the internal LAN.

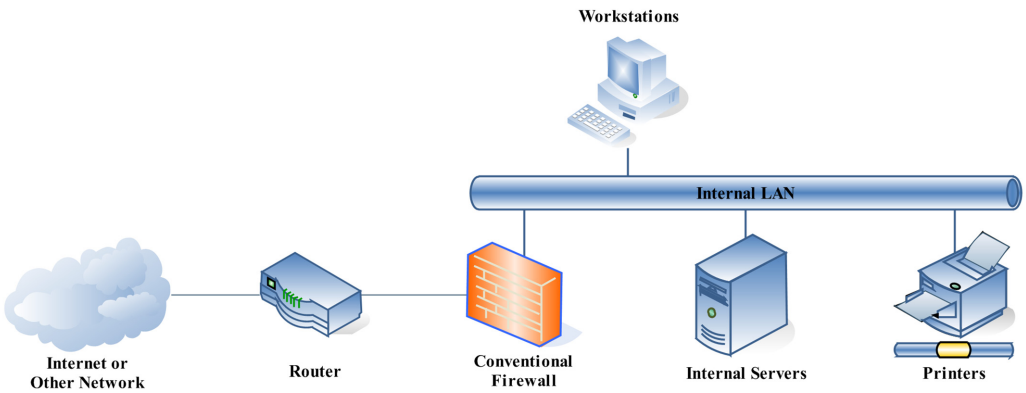

Figure 1.1. Typical placement of firewalls within early Internet-connected networks.

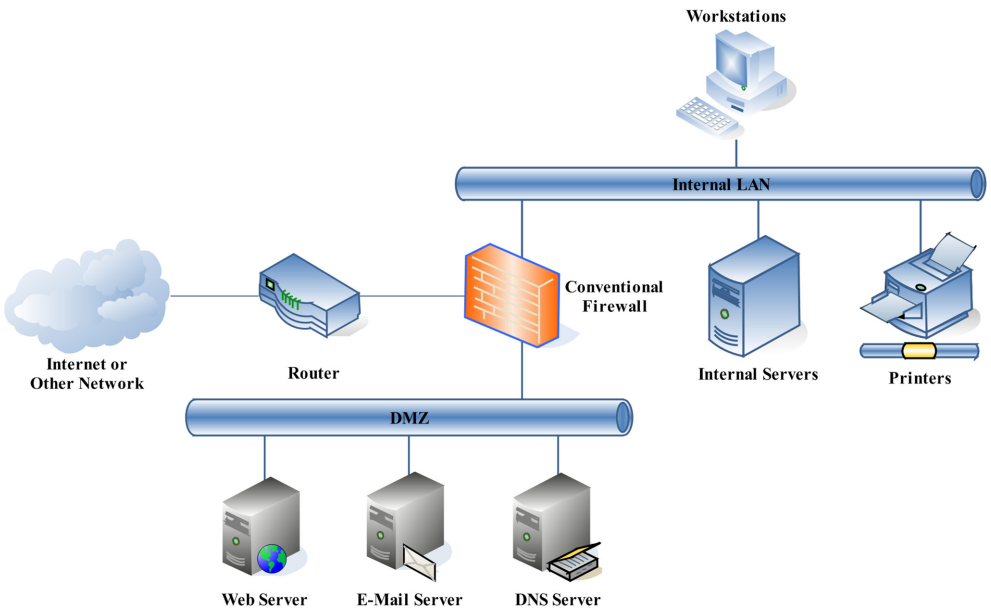

Figure 1.2. Creation of a DMZ using a single conventional firewall.

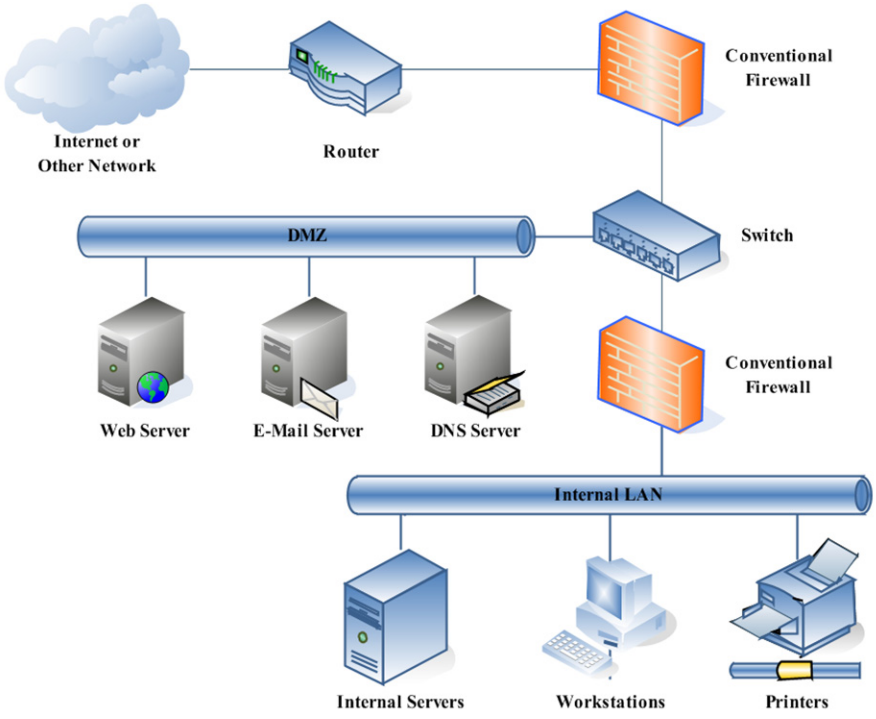

Figure 1.3. Creation of a DMZ using two conventional firewalls.

While more elaborate deployments of conventional firewalls exist, they are beyond the scope of this thesis. Instead, the fundamental properties of conventional firewalls and their placement in computer networks are key to the motivation for this research and thesis.

1.2.3. Shortcomings of Conventional Firewall Systems. Although tremendous advances in conventional firewall technology have been made, such progress has severely lagged against the raw number of services utilizing computer networks and the sophistication of attacks against such services that allow malicious entities to successfully exploit vulnerabilities against systems being protected by conventional firewalls. Despite even the creation of a theoretically perfect conventional firewall, such a system would be unable to protect its internal networks and their constituent systems from attack by virtue of the existence of fundamental flaws in the design of conventional firewalls. These design flaws are listed and described below, and proposed solutions are presented later.

1.2.3.1 Assumption of trustworthiness based on logical location. By default, conventional firewalls make assumptions about the trustworthiness of network devices—and ultimately the humans whose use those devices—based upon their logical location within the networks connected to them. Although a series of rules defining for each device IP address what level of trustworthiness to assume could in theory be established within a conventional firewall, implementing such a series of rules would be a painstaking feat for even a medium-sized network and would likely contain numerous misconfigurations that could be exploited by an attacker attempting to circumvent such restrictions. Even if a theoretically perfect set of such rules were to be developed, the ability to forge or 'spoof' IP addresses across LAN segments would make it trivial for a malicious entity to circumvent these rules.

The assumption that devices and users on the "inside" of a firewall are inherently trustworthy is dangerous. The estimate that roughly half of all attacks on internal systems are perpetrated by users who have been granted authorized access to the internal networks upon which those systems reside makes this danger evident [6]. Conventional firewalls by design are not able to address such insider threats, as the traffic from malicious entities whose target and point of presence are both on the same LAN is in no way monitored or controlled by such firewalls. Even malicious entities external to these

LANs can take advantage of this assumption by somehow coaxing internal systems or their users into initiating a connection to themselves or intermediary attack platform, launching an attack against the internal system using this connection, gaining control over the internal system, and subsequently using it to effectively establish a point of presence on the internal LAN.

1.2.3.2 Inability to police internal network traffic. Because conventional firewalls are deployed at network perimeters, they are only able to apply security policies to and alert on suspicious activity in traffic flowing between—and not within—networks. No capability to apply such policies to traffic whose source and destination are on the same network can therefore exist in conventional firewalls. This gap in the functionality of conventional firewalls is unfortunate again because there are no means through which to detect or prevent accidental or malicious actions against systems by entities that have legitimate access to networks upon which the targeted systems reside.

1.2.3.3 No protection against physical access to internal LANs. Although no system with strictly cyber elements will be able to physically prevent connections into internal LANs from occurring, such systems do have the potential to logically block access to networks and the data flowing across them from unauthorized physical connections into a network. Conventional firewalls are unable to provide such capabilities because their point of presence on the network is strictly at its border. Therefore, should a malicious entity gain physical access to a point of internal network ingress, their ability to obtain access to such networks is contingent only upon the absence of specialized network security hardware and software that prevents unauthorized physical connections. Such systems are not commonly deployed, and most commonly used ‘network logon’ packages do not prevent physically connected rogue systems from performing active traffic injection and passive network monitoring.

1.2.3.4 Circumvention by undocumented connections. The possibility exists that multiple means of ingress into internal networks may be present outside the knowledge or approval of the personnel responsible for managing such networks. Supposing that such undocumented and/or unauthorized connections exist, an attacker could utilize them to circumvent the protections of conventional firewalls given their point of presence would likely be within their trusted realms. Host-based defenses,

which are commonly either improperly maintained or unimplemented altogether, would then bear the sole responsibility of protecting internal systems from attacks launched via these connections.

As an example, suppose a user of a workstation that is connected to an internal LAN installs a modem in the workstation and, without notifying network personnel, connects the modem to the public telephone network in hopes of retrieving documents while traveling on business. This modem line could be discovered by an attacker performing reconnaissance against the organization and subsequently exploited by the attacker to gain access to the internal network without having to perform the risky task of penetrating the organization's Internet firewall.

The popularity of wireless networking poses other more probable scenarios. Users may for some reason install unauthorized and/or undocumented wireless networking equipment in internally networked devices in an effort to simplify their work, inadvertently providing an avenue for attack. Attackers could exploit these networked devices to gain direct high-speed access to corporate LANs, perfect for launching sophisticated attacks against other systems and stealing large amounts of data.

1.2.3.5 Disadvantages of chokepoint principle. Although the principle of acting as a chokepoint to the networks they are intended to protect is the primary enabler of the security services provided by conventional firewalls, this also works to their detriment. With the ever-increasing bandwidth requirements of modern computer operating systems and the applications that utilize Internet connectivity, the amount of traffic flowing through firewalls has risen significantly, bringing those firewalls closer to their theoretical maximum processing limits. And even should a firewall not be at its maximum throughput capacity, the more traffic that must be processed by a firewall per period of time, the more delay the firewall is likely to impose on that traffic when examining it for validity.

Aside from congestion issues, the chokepoint principle also makes conventional firewalls the focus of denial of service attacks, as they serve as a single point of connection for a network to "the rest of the world," relatively speaking. If it were the intent of a malicious entity to prevent an organization from accessing the Internet, for example, the firewall used to connect the organization to the Internet would likely be a

primary target of the attacker, as taking the conventional firewall out of service would ensure that traffic destined to the organization from the Internet would not arrive at its destination and vice versa.

1.2.4. History of the Distributed Firewall Concept. The concept of a distributed firewall, though not new but certainly in the computing world much younger than that of the conventional firewall, was publicly proposed by Steven M. Bellovin of AT&T Labs Research in 1999 [7]. In the proposal, Bellovin called for the policy enforcement role of the conventional firewall to be distributed throughout the endpoints of a network, such that each portion of the system perform policy enforcement on traffic that pertains to them rather than relying on a single system to do all such processing. Management of policies was to remain central, with communications between internally connected hosts secured via the IP Security (IPSec) network-level security mechanism for TCP/IP, developed as part of the IP version 6 protocol.

Bellovin's proposed system is indeed novel as it frees the concept of the conventional firewall from its dependence on its placement at the network border and the reliance on network topology for policy enforcement, arguably the most prevalent of its weaknesses. To further bolster the security of the distributed firewall system, Bellovin recommended that cryptographically signed certificates via IPSec be used as part of the distributed firewall concept to uniquely verify the identity of all networked devices before applying policies to traffic from them. Conventional firewalls and many other internal traffic-based security mechanisms rely on a combination of hostnames, data link layer (OSI layer 2), and network layer (OSI layer 3) addresses—all of which are easily forged—to identify devices on internal networks such that policies could be applied accordingly to the traffic.

Because manual administration of policies on each endpoint in a distributed firewall system would not be feasible for even medium-sized networks, Bellovin recommended that existing system management software, such as Microsoft System Management Server, be used to transfer policies from a central location to the endpoints.

In 2000, Bellovin and three colleagues from the University of Pennsylvania followed on his distributed firewall concept proposal with a paper outlining the implementation of such a distributed firewall in software for network hosts running the

OpenBSD operating system [8]. The software implementation was divided into three primary portions: a kernel-level portion that implemented enforcement mechanisms, a user-level daemon process to handle policies, and a device driver to facilitate bidirectional communication between the other two components. The four researchers utilized the KeyNote trust management system and its associated policy language for defining policies and distributing them from a central management server to all distributed firewall endpoints using a combination 'push-pull' client/server model. IPSec was utilized for user authentication, traffic protection, and credential distribution, as its network-level presence provides transparent security services to the layers above it in the OSI reference model, inclusive of applications interacting with the network via the application layer (Appendix A).

In 2001, with the financial support of the Defense Advanced Research Projects Agency, Charles Payne and Tom Markham of Secure Computing Corporation published [9] describing a hardware-based distributed firewall implementation they had developed. The duo had created a custom firmware image for the 3Com 3CR990 series of network interface cards (NICs) that would allow them to enforce firewall and other policies on traffic entering or leaving the NIC. A policy server centrally defined the policies enforced by each EFW-enabled NIC in the network. Payne and Markham claimed the system, dubbed the embedded distributed firewall or EFW, embraced a stronger protection model and was more resistant to tampering than the systems Bellovin and his three colleagues had proposed and implemented because they did not rely on the operating system of the network endpoint hosts and could prevent users of hosts from sniffing network traffic and launching attacks by means of address forgery.

Several other research endeavors have been undertaken on the topic of distributed firewalls, and a number of corporate-developed implementations have been released, such as the 3Com Embedded Firewall [10] (a commercially-available version of the concept developed by Payne and Markham), TheGreenBow Distributed Firewall [11], and most recently the Yoggie Gatekeeper Pro [12]. However, the concept of the distributed firewall—claimed near its inception to be of importance to the field of computer security—lost the interest of the mainstream research and development communities after 2001.

1.2.5. Securing Networks Using Distributed Firewall Systems. While each implementation of a distributed firewall operates in a slightly different manner, a generalized approach to securing networks using distributed firewalls exists. In such systems, rather than placing a conventional firewall only at the border between networks, the role of the conventional firewall is distributed amongst a collection of policy enforcement devices, typically under the control of a centralized system that is also used to define the policy enforced by the distributed firewall system, such that each enforcement component need only apply its policy to a single network endpoint. Figure 1.4 illustrates this generalized concept. Each miniaturized conventional firewall graphic overlaid onto a network endpoint indicates a form of policy enforcement device protecting the individual endpoint, either by software running on the endpoint or by a hardware device placed between the endpoint and the network, such as a NIC controlled by the distributed firewall controller/server.

In the case of Figure 1.4, a conventional firewall is maintained at the network border, despite the presence of a distributed firewall solution being deployed to protect each network endpoint. This configuration will be used throughout the remainder of this thesis and is both allowed and likely preferred in the majority of deployments, as it provides an additional layer of defense against attack from external entities and the conventional firewall can be configured such that it does not interfere with the operation of its distributed counterpart. In the case of a distributed firewall implemented in software, maintaining a conventional firewall at the border can provide some level of security to devices whose operating systems do not support the distributed firewall implementation or, in the case of a distributed firewall solution implemented into a NIC, devices whose hardware architecture do not support such a NIC.

1.2.6. Advantages of Distributed Firewalls Over Conventional Firewalls. The distributed firewall concept has many advantages over its conventional counterparts. These advantages are the byproduct of the omnipresence of the distributed firewall system throughout the network and the ability of such a system to uniquely identify its components.

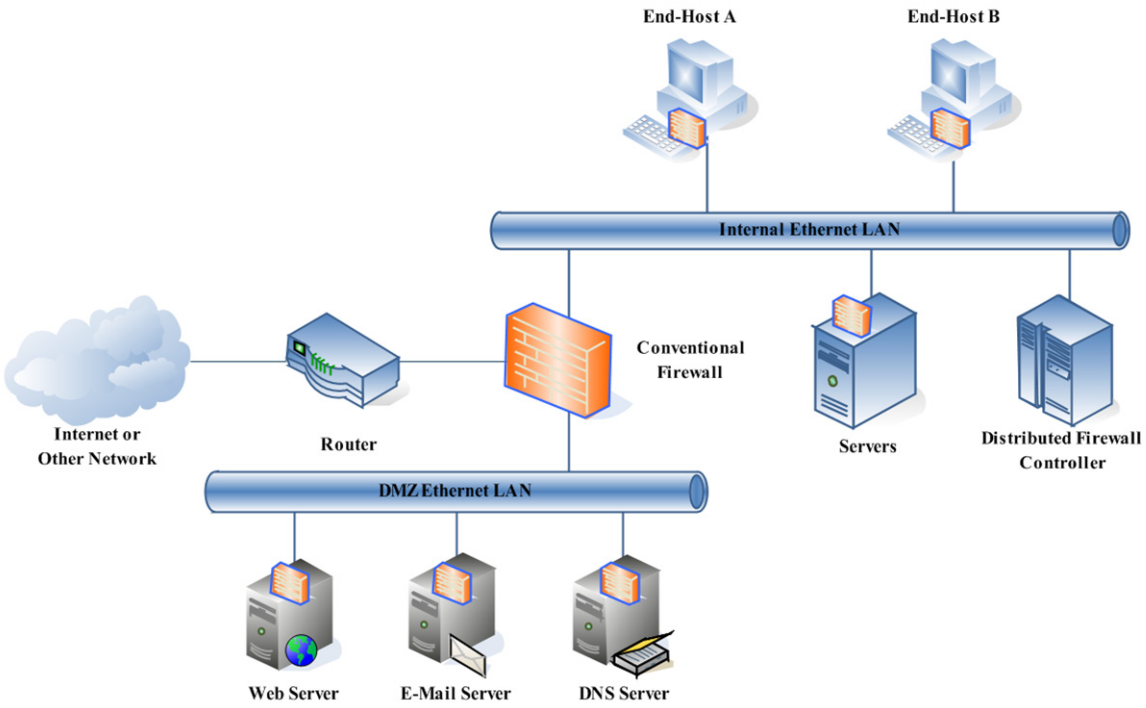

Figure 1.4. Generalized distributed firewall concept.

1.2.6.1 Lack of reliance on network topology. Because conventional firewalls are located at the border of networks, they are only able to assign definite levels of trust to entire networks rather than to the individual endpoint devices within such networks. Some flexibility in policy enforcement by conventional firewalls exists because IP addresses can be used within such policies; however, these IP addresses do not uniquely specify a particular device because such addresses can be arbitrarily assigned and hence easily spoofed. Ultimately, conventional firewalls must rely on network topology to ensure proper enforcement of policies. In distributed firewall systems, however, because each endpoint is separated from its network via an enforcement mechanism that acts as a part of the overall distributed firewall system, the possibility of uniquely identifying an endpoint exists, assuming the enforcement mechanism can be uniquely identified within the distributed firewall system.

Given the ability of a distributed firewall implementation to uniquely identify endpoints based upon means that are not easily spoofed, a tremendous flexibility in applying policies specific to particular network endpoints exists and can be exploited to

assign levels of trust to particular endpoints rather than by endpoints *en masse*. Therefore, if a host serves only a particular purpose, the distributed firewall enforcement mechanism for that host can examine all traffic destined to and from that host to ensure it meets the criteria associated with the function of the host. Appropriate traffic is allowed to traverse the enforcement mechanism, while all other traffic is denied.

Additionally, distributed firewall systems can provide the same level of protection to entities not physically located on internals networks as they do to internal systems. This capability is achievable because distributed firewalls do not rely on network topology, and no coarse "inside," "outside," and "DMZ" networks need be defined because policy enforcement occurs at the endpoint rather than at network boundaries. This attribute can be combined with existing virtual private networking (VPN) technology and is particularly useful to organizations that allow its members to telecommute via the Internet.

1.2.6.2 Ability to apply policy to all network traffic. Unlike the case in conventional firewalling, where only inter-network traffic traversing the firewall is subject to the defined firewall policy, a distributed firewall possess the ability to apply policies to all network traffic, regardless of its source or destination. This is by virtue of the firewall being distributed throughout the network rather than strictly at its border; therefore all traffic on the network—regardless of its source or destination—is placed under the scrutiny of a policy defined for an enforcement mechanism in the distributed firewall system.

Under conventional firewalling, should an external malicious entity penetrate the firewall by some means, s/he would have unobstructed access to the internal systems protected by the firewall. This is not the case where a distributed firewall is deployed, as traffic even between two hosts connected to the same LAN is subjected to at least two instances of a policy defined for the distributed firewall system.

1.2.6.3 Addressing of internal threats. Because distributed firewalls have the ability to apply enforcement policies to all network traffic regardless of its point of origin, there exists the ability to address 'insider threats,' or threats perpetrated by entities that already for some reason have some legitimate form of access to systems that are connected to what would be the 'inside' of conventional firewalls. With a distributed

firewall solution deployed, the feat of gaining access—either legitimately as an employee with authorized access to the network containing the target system or illegitimately as an attacker that has in some way gained control over an internal system—to the internal network affords an attacker only a portion of their goal in accessing other internal systems because the policies of the distributed firewall should be constructed and enforced to thwart such activities.

The ability of distributed firewalls to address insider threats is important, as it has been found that nearly half of all attacks against internal systems have been perpetrated by those that are 'insiders' possessing some level of authorized access to the systems they are attacking [6]. While disgruntled members of organizations may intentionally launch these attacks or well-meaning members accidentally access systems in a manner that is not expressly intended, a properly implemented distributed firewall has the ability to address such threats.

1.2.6.4 Flexibility in functionality. The role of the conventional firewall can be extended beyond the enforcement of policies on network traffic. Other services, such as intrusion detection and prevention, event auditing, bandwidth limitation, and remote user authentication can also be performed by conventional firewalls. These services, however, have limited scope due to their placement.

The possibilities for services beyond network defense that exist for distributed firewall systems are nearly endless. Because of their distributed nature, such systems potentially are able to provide a variety of services that can bolster network security. The ability to form firewalling rules for each user on a network, rather than each host, exists, as does distributed intrusion detection and response, event auditing, and other advanced functions. Another such service proposed in [9] is similar to virtual private networking but, rather than two endpoints communicating via an encrypted tunnel, an entire subset of a network does so. Many other useful features have been proposed and it is likely that a plethora of others have yet to be conceived.

1.2.6.5 Robustness against denial of service attacks. As discussed, one disadvantage of conventional firewalls is that their being a chokepoint often makes them a target in denial of service attacks. Because a distributed firewall does not act as a chokepoint to the internal networks they protect, there is no single entity within a

distributed firewall system that an attacker can target to orchestrate a denial of service attack against an entire network. Instead, an attacker must focus their efforts against all of the enforcement mechanisms of the distributed firewall, a subset of these mechanisms, or the controller/server that coordinates the actions of these enforcement mechanisms. The division of firewalling tasks amongst a group of devices makes the perpetration of a denial of service attack much more difficult, as a large group of targets must be successfully burdened before an effect in performance is seen. Attacking the controller/server of distributed firewall systems results only in limited success because implementations of such systems should address the possibility that the controller/server portion be unavailable for various periods of time by operating autonomously and/or via a backup controller/server.

1.2.7. Shortcomings of Existing Distributed Firewall Systems. Although generalizations between implementations of distributed firewall systems exists, many variations between such implementations are also present. The two implementations referenced previously in this thesis—[8] and [9]—are certainly amongst the most popular of the few existing implementations of distributed firewalls and they vary significantly. The implementation in [8] exists strictly in software, while that in [9] exists in a combination of software and hardware. Each implementation relies on different features, cryptographic methods, and host system resources to achieve its purpose and each has its disadvantages. Amongst the implementations researched for this thesis, serious shortcomings exist that could have been avoided.

In [8], the researchers chose to integrate their distributed firewall technology into both the kernel and user spaces of the OpenBSD operating system. By intercepting calls to two network library functions applications use in communication with the network—*connect* and *accept*—the firewall software is able to examine connections to and from the host machine and, after applying a central policy defined for the firewall system, allow or deny these connections.

While indeed novel, this implementation suffers from numerous shortcomings. First, those applications that are not linked against the version of the network library containing the distributed firewall functions can simply bypass enforcement altogether. This disrupts the ability of the system to enforce policies on all traffic within the network

and can allow attackers that have somehow compromised an internal host to establish rogue connections to and from the compromised host.

Another issue with implementing the distributed firewall in software is that it may be possible for malicious users of hosts running the software to disable such protections or tamper with the resources assigned to the software to change its behavior. The ability to tamper with such resources may reveal vulnerabilities in the software that can be exploited remotely by attackers to gain access to systems running such implementations.

Lastly, because this implementation and likely all software implementations of distributed firewalls will require modification to operating system software and configurations, such systems will very likely not be operating system independent. This is of concern because a wide range of operating systems are run on the many thousands of models and types of networked devices. Therefore, these implementations would have to be ported to numerous operating systems, each of which handle networking functionality in a different manner, and some of which may be unable to support these capabilities due to limited resources or other such factors.

In [9], the functionality of the distributed firewall system proposed was implemented into the NIC of devices to be policed by the distributed firewall. A customized firmware was written for the 3Com 3CR990 series of NICs that allowed them to examine traffic flowing in and out and allow or deny such traffic based upon policies sent to the NIC by a central policy server. Because of the limited processing and storage capabilities of the NIC, some interaction with the device operating system took place, namely in the form of a “helper agent” that periodically sent a heartbeat signal to the policy server and provided to the NIC the IP address(es) bound to the interface.

Although separating the endpoint distributed firewall functionality from the host operating system is a significant advance in securing the implementation from attack by potentially malicious users of the host, the implementation still relies on the resources of the host and its operating system, both of which may in some way provide attack vectors into the system. In addition, the very limited processing and storage capabilities of the modified NICs limits the flexibility of the system to provide strong cryptographic support and advanced security services such as those previously discussed.

Supposing that the helper agent could be reverse engineered, it may be possible for a malicious version of the agent to be written that provides the NIC with false information but produces valid heartbeat signals to the policy server. Worse, a malicious application for the host could be engineered such that a generic NIC could be used to replace the modified 3CR990 without the knowledge of the policy server.

Most problematic with this implementation is that it must rely on symmetric key cryptography to secure communications between the policy server and the distributed firewall system endpoints given the limited resources of the modified NICs. This means that should a malicious attacker gain knowledge of the key used to secure this communication—and this key must somewhere be stored in the permanent memory of all firewall-enabled NICs and the policy server—the attacker could easily decrypt traffic sent via the network between the endpoints and the policy server and could likely send commands and policies to the endpoints posing as the policy server.

As with the implementation in [8], the presence of a helper function that must be run by the device operating system implies that this implementation is also not operating system independent. Additionally, not all devices connected to a network may have an expansion slot to support a NIC, rendering the device incapable of being protected by this particular distributed firewall implementation.

1.3. PROPOSED SOLUTION

The concept of the distributed firewall has great potential, despite its brief and limited appearance in the realm of computer and network security research and development since its inception in 1999. Two implementations of a distributed firewall have been previously discussed and their shortcomings analyzed in this thesis. To solve many of these shortcomings, this thesis proposes that the enforcement mechanisms of distributed firewall systems operate completely autonomously from the endpoint devices they are intended to protect.

This thesis was produced in conjunction with a research and proof-of-concept implementation development endeavor, to which the remainder of this thesis is dedicated. In this research and development, a general distributed network security framework is

proposed and developed, and a distributed firewall system is given as an application of that framework.

1.3.1. Hardened Distributed Network Endpoint Security System Framework. The general distributed network security system that is proposed and developed as part of the research endeavor associated with this thesis is intended to serve as a framework from which any distributed network application within the hardware and software limitations of the nodes and controllers can be based. This framework, dubbed the Hardened Distributed Network Endpoint Security System (HarDNESS), consists of an arbitrary number of nodes distributed throughout an Ethernet network utilizing the TCP/IP protocol suite and one or more controllers, or servers, whose duty it is to manage these nodes such that the tasks of an application utilizing the framework can be achieved at each network endpoint where a node exists. An emphasis on security has been taken in the design and implementation of this framework, hence its description as a hardened system.

1.3.2. Application of Framework to Distributed Firewall Systems. The second portion of the research and development associated with this thesis is in the form of an application of the HarDNESS framework toward the creation of a distributed firewall solution named the HarDNESS Distributed Firewall Application (HarDNESS DFA). This solution provides similar distributed firewall functionality as the two previously discussed distributed firewall implementations while addressing their shortcomings, as outlined earlier in this thesis.

To implement HarDNESS as a distributed firewall system, each node, referred to as a distributed firewall node (DFN), is a Linksys WRT54G Hardware Revision 3.0 wireless router that runs the customized HarDNESS firmware. The firmware in its current proof-of-concept state entirely disables the wireless functionality of the DFNs, in essence making them five-port routers supporting 10Mbps or 100Mbps Ethernet over unshielded twisted pair media. Included in the firmware image are the components necessary to allow each DFN to examine and filter traffic flowing between any of its five Ethernet ports, namely the *ebtables* and *iptables* filtering suites (Section 2.3.1).

2. APPROACH AND METHODOLOGY

2.1. DISTRIBUTED SYSTEM FRAMEWORK DESIGN

This thesis proposes a distributed network security system framework, called the Hardened Distributed Network Endpoint Security System (HarDNESS), whose purpose is to provide a foundation for distributed network security applications while avoiding the issues of similar systems that have been previously addressed in this thesis. This section provides an overview of the HarDNESS framework and addresses its design.

2.1.1. Framework Overview. Two categories of hardware components form the basis of the HarDNESS framework—nodes and controllers. Nodes are lightweight general-purpose computing devices with at least three logically distinguishable 10/100Mbps auto-negotiation Ethernet interfaces capable of running embedded or reduced distributions of the Linux operating system. Nodes are physically placed between network endpoints requiring the security services provided by HarDNESS applications and the network itself, hence having complete access to the traffic flowing between the network and the protected endpoint device.

To coordinate the collective and individual behaviors of nodes and provide a central point of system administration, the other component within the HarDNESS framework, the controller, is deployed on an out-of-band network to which all nodes under the controller are also connected. Multiple controllers may be deployed to accommodate large numbers of nodes and to provide continuous operations in the event other controllers becoming unavailable. A secure communications channel is then established between each node and its primary controller so that command and control (C2) and other HarDNESS application communications can take place between them. In the event that the primary controller for a node is unavailable, the node may establish a secure C2 channel with a specified backup controller until its primary controller becomes available.

2.1.2. Design Factors. In the design of the HarDNESS framework, several factors were identified as being of key concern to the development of the HarDNESS framework. These factors were determined according to the nature of the distributed system and the types of distributed applications that would be likely to implement the

framework as a foundation. These factors are given here in ascending order of their priority in the design of the framework system.

2.1.2.1 Security. Because the HarDNESS framework is primarily intended to provide a foundation for distributed network security applications, the paramount factor in its design is security. This is necessary as it is likely that, because the services provided by applications implementing HarDNESS will likely be geared toward security, such applications will be a prime target for attack.

The first means by which the system is secured is via the protection of all communications between all framework components. This protection is provided by two primary means, the first of which is through separation of the C2 communication traffic between components of HarDNESS and that of the network being protected. This is achieved via either logical or physical separation of C2 traffic. These separations are detailed further in Section 2.1.3. The other means by which inter-component traffic is protected is via the establishment of encrypted tunnels between the nodes and controllers distributed throughout the network. These tunnels are created by use of the OpenVPN client/server software package, which is further discussed in Section 2.2.2.2.

Other security principles are incorporated into the design of the HarDNESS framework. These principles deal with the manner in which applications and services on the components of the framework are deployed and operated. The first of these principles is that, whenever possible, the smallest number of instances of a particular service is run across the distributed system. This is important because in a distributed system it is much more difficult to properly manage and secure large numbers of service instances run across the collection of components rather than maintaining only a small number of such instances. In relation to HarDNESS, this principle equates to, whenever possible, running instances of a service only on the controllers within the distributed system, rather than running such on every node, as in large deployments the number of nodes will be much larger than the number of controllers. This also adds the advantages of keeping free the limited resources of the nodes and simplifying monitoring and management of the overall distributed system.

The other principle adopted in the design of the framework is to run the smallest number of applications and services necessary to support the operation of the system.

This reduces the number of possible attack vectors into the system, keeps the resources across the distributed system as free as possible, and streamlines proper setup and management of the system.

2.1.2.2 Scalability. Modern networks may contain hundreds and even thousands of endpoint devices, each of which potentially can be used as a vector for penetrating the security of the entire network. Therefore, security systems for these networks must be able to extend their services to very large numbers of network devices. The issue of scalability was therefore a top priority in the design of the HarDNESS framework.

To address the issue of scalability, focus was placed on the controllers, as increasing the number of nodes in the system places burden only on the controllers within the distributed system and the network tasked with transporting the additional C2 traffic to support these nodes. Given that the amount of C2 traffic between controllers and nodes is minimal and that standard 100Mbps Ethernet LANs should easily be able to support such traffic, this case was not considered to be a retardant to scalability. However, resources on the controllers required for each node can be significant depending upon the application utilizing the HarDNESS framework, and therefore efforts were made to keep the amount of controller resources per connected node as small as possible in the case of framework operations.

2.1.2.3 Robustness and reliability. Networks are often central to the operations of the organizations that implement them. Many times systems that are connected to those networks must demonstrate some sort of measure for reliability. Although no formal measures for reliability have been made for the HarDNESS framework, robustness and reliability were taken into account in its design.

To increase reliability and robustness in the HarDNESS distributed framework, focus was placed on the components within the system. At the nodes, running the smallest subset of applications and services possible to support the operations of the framework decreases the possibility of software errors. Additionally, the framework allows administrators to create customized scripts to deal with events such as the primary controller for the node becoming unavailable during its operation and the assignment of backup controllers to each node.

It is advocated that controllers be used only to support the operation of the distributed application(s) utilizing the HarDNESS framework so as to prevent problems in compatibility and reliability. The principle of requiring a small subset of minimal, hardened services and applications on controllers was adopted, just as with on the nodes, in hopes of increasing the security and reliability of the controllers.

2.1.2.4 Transparency. Many novel security systems, both in software and hardware form, have either failed to be adopted or have been intentionally bypassed simply because they are considered too invasive to the users and processes they are intended to protect. One design goal for the HarDNESS framework is to provide a foundation for distributed security applications that are in the best case completely transparent to the users and processes of network endpoint devices, and in the worse case present the least level of invasion as possible.

To keep from being invasive to the network endpoint devices they are protecting, nodes within HarDNESS deployments are completely separated, both by software and hardware means, from the actual network endpoint devices they are protecting. This reduces invasion to users and processes because the software systems and configurations of the endpoints need not be changed to enable HarDNESS applications to properly function. Additionally, separating nodes from endpoint devices allows them to provide security services to devices that may be unable to implement other distributed security systems requiring software and hardware modifications to the devices, such as the two distributed firewall implementations discussed previously in this thesis.

2.1.2.5 Customizability. Because HarDNESS is a framework upon which distributed applications can be built, it is designed to maximize customizability. Whenever possible, as few assumptions as possible are made as to the applications that would implement the HarDNESS framework, allowing the framework to be flexible and support a wide range of applications.

2.1.3. Command and Control Communications Architecture. To control the behavior of the collection of nodes throughout the network, one or more controller hosts must be configured and established on either a truly- or semi-out-of-band command and control (C2) network to which all nodes are also connected. Placing the controller(s) and nodes on a truly-out-of-band C2 network, that is, a physically separate network from the

one being protected by the HarDNESS framework, increases the security of the overall system, as gaining access to the out-of-band control network via attack on the nodes has been made difficult. Deployments utilizing a truly-out-of-band C2 network should prevent physical access to that network from being easily obtained; therefore, it is recommended that the nodes be placed in secured locations, such as limited-access wire closets, rather than in locations where physical access to them is easily obtained. The case where a truly-out-of-band C2 network is utilized is illustrated in Figure 2.1.

Because creating an entirely separate physical network for the purpose of HarDNESS command and control is cost prohibitive to most organizations, a semi-out-of-band C2 network may instead be utilized. A semi-out-of-band C2 network configuration is considered to be one where the C2 network is constructed using the same physical network infrastructure as the network being protected by HarDNESS, but logically the traffic on these two networks is separated. This is achievable by using a different IP address range for the two networks and by exploiting traffic encryption mechanisms to protect the confidentiality and integrity of C2 traffic. The case where a semi-out-of-band network is utilized is illustrated in Figure 2.2.

Upon booting up, each node establishes a secure C2 channel with the primary controller from which it has been configured to accept commands. Through this encrypted channel, controllers are able to influence the behavior of individual nodes throughout the entire network, in essence making the controllers central points of control over the distributed system. The controllers host support services to the nodes through these secure C2 channels so as to harden the system against attack. Additionally, all status information relayed to the controllers by the nodes is transmitted via the secure C2 channel so as to protect its integrity and confidentiality while in transit.

2.2. PROOF-OF-CONCEPT FRAMEWORK IMPLEMENTATION

To demonstrate the feasibility of the proposed HarDNESS framework, a proof-of-concept implementation of the proposed system was developed as part of the research associated with this thesis. Details on this proof-of-concept implementation are given in the following sections.

2.2.1. Node Hardware Platform. The nodes within the proof-of-concept HarDNESS framework consist of Linksys WRT54G wireless routers running a customized firmware. In an effort to capture a share in the growing wireless networking market, Linksys, now a subsidiary of Cisco Systems, released in early 2003 the first version of their now pervasive WRT54G wireless broadband router. Its low cost and impressive set of features made it a dominator in the home and small office networking market [13].

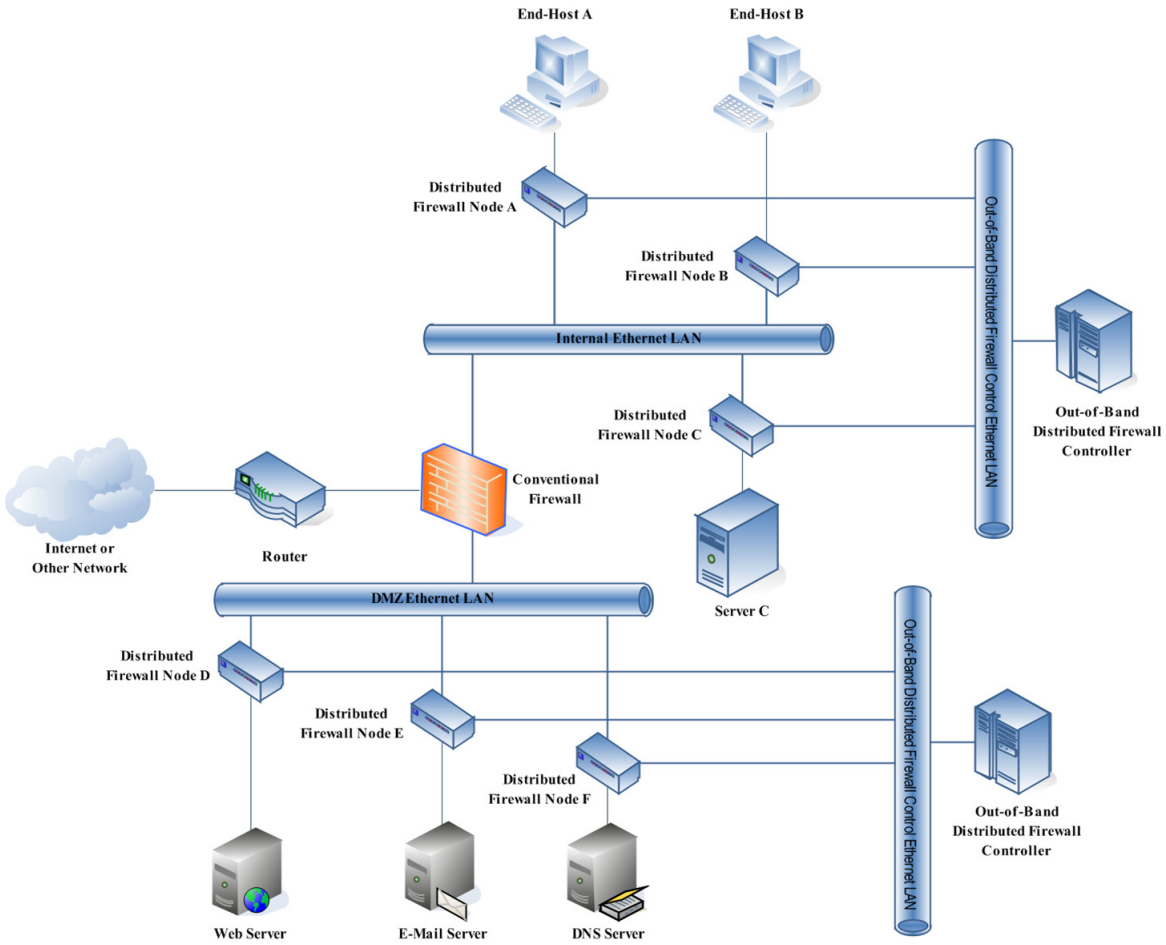

Figure 2.1. HarDNESS framework utilizing a truly-out-of-band C2 network.

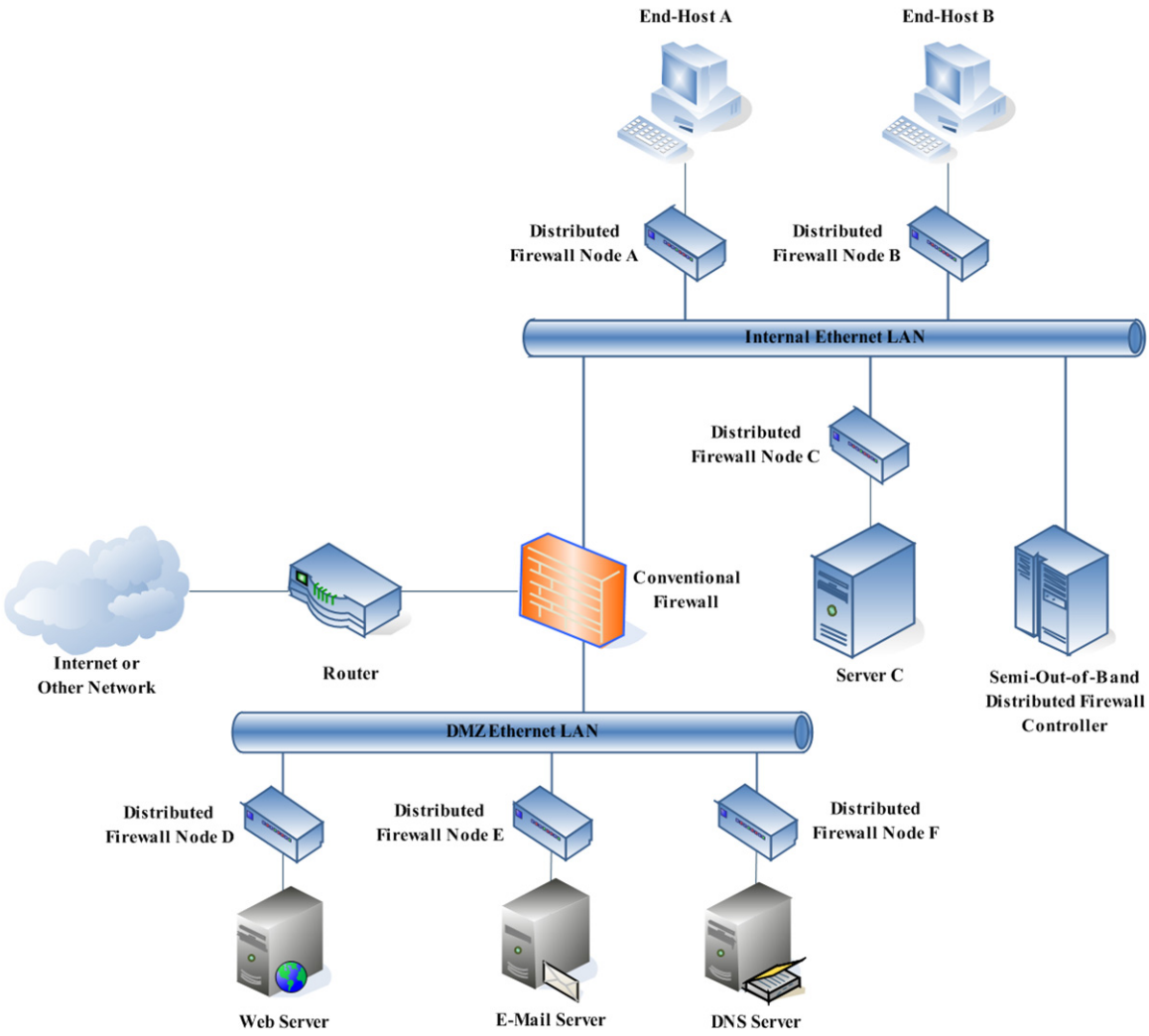

Figure 2.2. HarDNESS utilizing a semi-out-of-band C2 network.

As a consumer device, Linksys designed the WRT54G series of wireless routers with several key requirements. The device utilized as little dedicated hardware as possible so as to reduce its cost of manufacture, hence requiring the bulk of its functionality to be performed via software systems. As another measure of reducing costs, the firmware for the WRT54G router series was based off of the free-for-use Linux operating system. This reliance on Linux then required Linksys to release the source code for the firmware to the public at no charge due to the terms of the GNU Public License (GPL). The Linux design decision, unbeknownst to Linksys at the time, has since had a considerable effect on the entire routing industry, as it ushered in the ability to transform

a sub-US$100 router into one with features of its US$600 cousins, all through software-only means [14].

Since the release of the first model of WRT54G broadband wireless router in 2003, Linksys has produced several variants. These variants exist in the form of different hardware revisions, increased maximum wireless throughput capabilities, and most recently versions whose firmwares are not based upon Linux. Most notable in the context of this thesis are the variants' abilities to run common third-party firmware released for the WRT54G series. Because the proof-of-concept HarDNESS framework uses the Linksys WRT54G router in a manner for which it was not originally designed, the ability of the hardware version utilized in the proof-of-concept framework to run a third-party firmware is critical. Therefore, in the proof-of-concept framework implementation, hardware revision 3.0 of the Linksys WRT54G router was chosen as the node hardware platform, as that version provides the most impressive hardware set and is supported by the vast majority of third-party firmware packages, including the package selected as the backbone for HarDNESS, OpenWrt White Russian Release Candidate 6 [15].

Because Linksys designed the WRT54G series of broadband routers to be an inexpensive consumer product that relies as little upon dedicated hardware as possible, these routers feature relatively impressive general hardware specifications. Hardware revision 3.0 of the WRT54G router features the following: a Broadcom BCM4712KPB combined central processing unit and media access control chip running at 200MHz, 4MB of flash (writable, nonvolatile) storage, 16MB of RAM, a wireless Ethernet controller, and a programmable 6-port Ethernet switch. The hardware architecture of the Broadcom BCM4712KPB CPU is MIPSEL, or MIPS with little endian byte order.

The programmable 6-port Ethernet switch included with the Linksys WRT54G supports virtual LAN (VLAN) assignments to each port. In essence, VLAN support allows a user to configure a set of one or more ports on the switch independently of the others, hence allowing the multi-port switch to create unique, segregated "virtual LANs." In addition, each VLAN configured appears as a standard Ethernet interface to the operating system; thus, a user can configure each VLAN interface with unique Ethernet media access controller (MAC) and IP addresses. Routing and firewalling between VLANs is also possible by use of the *iptables* utility, which is detailed further in Section

2.3.2. For example, the combination of VLAN support and *iptables* is used by Linksys in configuring the Ethernet devices within the WRT54G and segregating traffic between the Internet/WAN and LAN sides of the router.

2.2.2. Software Subsystems. A variety of software subsystems are required on the nodes and controllers of HarDNESS deployments to provide a basis of support for the distributed network applications that utilize the HarDNESS framework. The most central of these software subsystems are detailed here.

2.2.2.1 OpenWrt firmware and software packages. The primary software package involved in the operation of the nodes in the proof-of-concept HarDNESS framework is the OpenWrt firmware package. This firmware package is an open source third-party solution for a variety of consumer wireless routers that provides an embedded (reduced) Linux kernel and operating system, along with a number of required and optional software packages to provide advanced networking and routing functionality.

The decision to use the OpenWrt firmware over its numerous peers is natural given its modularity and the ease in which applications and services not included in the original OpenWrt distribution can be added. With the exception of the knock daemon, which is discussed in Section 2.2.2.3, OpenWrt White Russian Release Candidate 6 provides all the software packages necessary to implement the proof-of-concept HarDNESS on the Linksys WRT54G hardware revision 3.0 router. Some modifications to the software packages are required, however, but are easily implemented via the capability within the OpenWrt software development kit to apply patches to the source code of software packages integrated into the distribution.

Included as a part of the OpenWrt firmware is the ability to perform firewalling, routing, and network address translation (NAT) on traffic traversing any physical or virtual Ethernet interface that has been assigned an IP address. With the addition of another package to the base OpenWrt firmware that is discussed in Section 2.3.3, it is found that the binding of an IP address to the interfaces to be controlled via firewall policies is not a prerequisite, opening the possibilities of advanced routing and firewalling functionality.

2.2.2.2 OpenVPN. Securing C2 communications is performed via the use of OpenVPN, a flexible and full-featured Secure Sockets Layer (SSL) Virtual Private Networking (VPN) solution [16]. An instance of an OpenVPN server is run on each controller, and as part of its boot procedure, each node attempts to establish an encrypted VPN tunnel with its primary controller via the use of the OpenVPN client. Once established, this encrypted tunnel serves as the secure C2 communications channel between the node and its controller, with all traffic between the node and its controller flowing through it. When traffic between a node and a controller is sent, it is first transparently encrypted by the OpenVPN software, and when it is received at the other tunnel endpoint, it is transparently decrypted by the OpenVPN software before being sent on to the appropriate application or service to which the traffic was originally destined.

Although OpenVPN provides an enormous variety of configuration, authentication, and traffic security options, only a small subset of these options are required by the HarDNESS framework. Because the primary design principle for HarDNESS is to maintain its security, the configuration for OpenVPN servers and clients within the HarDNESS framework is such that security is paramount. For example, public key infrastructure (PKI) mechanisms within OpenVPN are utilized in the authentication procedures that take place when nodes connect to controllers. Although detailed discussion of PKI concepts is beyond the scope of this thesis, the use of PKI authentication is important because, unlike under most other authentication systems, not only does the controller authenticate the node, the node also authenticates the controller. This is important in preventing the spoofing of the identity of a node and gaining some access to a controller and also in preventing the spoofing of the identity of a controller to gain C2 access to nodes.

Because the PKI mechanisms within OpenVPN require that the system time kept by the client and the server be synchronized to within a certain differential, it is necessary that nodes first synchronize their system time with that of their controller before attempting to create a VPN tunnel. To perform this synchronization, the low-risk Network Time Protocol (NTP) is used—an NTP server is run on controllers and is accessible to all nodes prior to their forming a secure C2 channel, and an NTP client on

each node performs a request for the system time from its controller and uses the replies to synchronize its clock with that of its controller.

Instances of OpenVPN within HarDNESS deployments are configured to ignore traffic sent to the ports used by OpenVPN unless such traffic is cryptographically verified as being authentically generated by a node or controller in the distributed system. This is critical to thwarting denial of service attacks that may be launched against the OpenVPN subsystems present within HarDNESS deployments, as well as successfully stopping reconnaissance attempts from discovering the presence of running OpenVPN servers on controllers.

2.2.2.3 Knock daemon and client. Because the nodes in HarDNESS deployments are configured to not run superfluous services, instructing a node to execute commands specified by a controller is done in a reverse client-server model. To instruct a node to initiate retrieval of commands from its controller, a "port knock" daemon is run on the node that listens for events generated by a port knock client on the controller [17]. A port knock daemon is a software application that promiscuously monitors one or more network interfaces for a series of "knocks" on a specific sequence of TCP and/or UDP ports within a specified time frame that has been generated by a port knock client. If a knock sequence matches one of those specified in the configuration file for the daemon, the daemon executes the command associated with the sequence, which is also provided via the configuration file. Such can consist of any command specifiable at the command line, i.e., the execution of a program, running of a script, setting of an environment variable, etc.

The use of the knock daemon for having nodes contact their controller was chosen for two primary reasons. First, running a service that listens for command execution requests from a controller could be discovered and exploited should a malicious entity gain access to the secure C2 communications channel established between nodes and controllers. The knock daemon does not open any ports when listening for knock sequences, but rather listens promiscuously to network traffic on one or more interfaces at the data link layer (Appendix A) to determine if valid port knock sequences have occurred. This provides a means of stealth against reconnaissance attempts on nodes.

The second reason for which a knock daemon was chosen is that the knock sequences used to initiate the execution of commands on the nodes can be made random and synchronized with the controller for a node. Suppose, for example, that fixed port knock sequences were in use, i.e., a valid sequence consisted of a knock on TCP port 7000, followed by a knock on TCP port 8000, and concluded with a knock on TCP port 9000, with all knocks having the TCP SYN (synchronize) flag set. An entity monitoring the communications channel through which these knocks were generated could easily perform a replay attack by simply reissuing the knock sequence. The knock daemon utilized in HarDNESS possesses the capability to specify knock sequences for a command initiation in the form of a "one-time pad" file that can be randomly generated and synchronized with the host expected to issue knock sequences. As each sequence in the file is used, both the knock issuer and daemon remove the used sequences from the one-time pad file as they are issued and received, respectively. Therefore, each sequence in the one-time pad file is used only once, effectively protecting the daemon from replay attacks.

Because the use of one-time pad sequence files can potentially result in the loss of synchronization between the pad file used by the daemon and that used by the client, HarDNESS nodes utilize a fixed knock sequence whose sole purpose is to have nodes resynchronize their one-time pad files with the controller. While seemingly this fixed sequence could be discovered and used in the orchestration of attacks against nodes, nodes have been instructed to, upon receiving this fixed sequence, receive the updated one-time pad file from only their controller via an additionally secured tunnel encapsulated within the secure C2 communications channel. This encapsulated secured tunnel is created via the use of the Secure Shell (SSH) protocol, which is discussed in the following section.

2.2.2.4 OpenSSH. As a means of further securing the transfer of C2 information between nodes and controllers, a free and open source implementation of Secure Shell, called OpenSSH, is used to create encrypted communication channels within the secure C2 communication channel provided by OpenVPN to transfer command scripts from controllers to nodes. OpenSSH offers an enormously flexible, secure communications implementation that has been very widely adopted throughout the

field of computer networking. The OpenSSH suite provides replacement versions of the *ftp*, *telnet*, *rlogin*, and *rcp* command-line tools that have been hardened, utilize encryption on all traffic generated, and implement a variety of advanced authentication schemes [18].

Although the OpenSSH suite is primarily used for secure remote terminal access between hosts connected by public or otherwise untrusted networks, its use within the HarDNESS framework is to provide an additionally secured means of transferring command scripts to nodes from their controllers. A particular utility within the OpenSSH suite, Secure Copy (SCP), performs these transfers.

The authentication mechanism used in the establishment of SSH connections between nodes and controllers is a reduced version of the public key infrastructure mechanism used by OpenVPN and is commonly referred to as public key authentication. Under this scheme, each node and each controller will possess two cryptographically related keys, one referred to as a "public" key that can be distributed without impact to security, and a "private" key that is to remain known only to the entity to which the public/private key pair is associated. Despite their cryptographic relation, deducing the private key knowing only the public key has been made incredibly difficult. The public keys for all nodes are distributed to all controllers intended to control those nodes, and the public keys of all controllers are distributed to all nodes to which the controller will provide services. The private keys for all nodes and controllers are stored only on the device to which the public/private key pair has been assigned such that access to the private key is as heavily restricted as possible. Because these two keys are cryptographically related, the node can challenge the controller on its knowledge of the private key associated with its public key, and vice versa. In this sense, just as with the PKI authentication mechanisms, nodes authenticate the identity of controllers to which they are attempting to connect, and the controllers also authenticate the identity of the nodes attempting to connect to them.

2.2.2.5 HTTP daemon and configuration interface. Because a large number of nodes may be deployed in networks, the initial configuration for nodes must be made as simple as possible. To streamline this process, nodes that have yet to be configured run a Hypertext Transfer Protocol (HTTP) daemon to which an administrator can connect

using a standard web browser. The forms served via this interface contain basic HarDNESS framework configuration settings that can be modified according to the function of the node, such as its name and group assignment, the hostname of the controller under which the node is to function, and the cryptographic credentials necessary for the node to establish a secure C2 communications channel with its controller. The configuration interface is customizable so as to accommodate HarDNESS applications that require configuration options to be specified on nodes prior to their deployment.

Connections to the HTTP configuration interface can be made only after a fixed knock sequence is issued to an unconfigured node. After connecting to the interface via a web browser, it is advised that administrators fully configure the node, replacing the default configuration settings, which in turn disables the running of the HTTP configuration interface in subsequent boots of the node. Configuration changes made after deployment are manually performed by issuing commands to the node from its controller.

2.3. PROOF-OF-CONCEPT DISTRIBUTED FIREWALL APPLICATION

As a portion of the research associated with this thesis, a proof-of-concept distributed firewall application, dubbed the HarDNESS Distributed Firewall Application (HarDNESS DFA), was developed from the proof-of-concept HarDNESS framework described in Section **Error! Reference source not found.**. The additions required to the proof-of-concept HarDNESS framework are minimal, as the development of a distributed firewall lends itself naturally as an application of the HarDNESS framework. The support functionalities provided by the HarDNESS framework allow for command and control of the distributed firewall nodes (DFNs), including the transfer of enforcement policies from controllers to the DFNs. The additional software packages added to the nodes to provide for the filtering of network traffic are described in the following sections.

2.3.1. Transparent Firewalling Capability. On "vanilla" WRT54G routers running the OpenWrt firmware, firewalling is possible but is performed under the assumption that network address translation (NAT) is used. Under NAT, the 'internal' network is assigned a private IP address range (or a range of IP addresses that cannot be routed via the Internet and are hence reserved for internal use) and a single Internet-routable IP address is assigned to the Internet-facing interface of the router. The router then translates between the private addresses assigned to internal devices and that of the Internet-facing interface whenever Internet access is required; thus enabling Internet access to all internal devices through a single routable Internet address.

Because NAT is only semi-transparent to network applications, using NAT in the HarDNESS DFA is considered to be too invasive as the DFA may be deployed on networks where NAT-incompatible applications are in use. A means of 'transparently' applying firewall policies to traffic flowing between the network and its endpoints was sought instead. The difficulty in transparent firewalling lies in the fact that the firewall device—in this case, the DFNs—must apply policies to traffic that is not destined for the firewall device itself. This difficulty was overcome using two software packages, *ebtables* and *iptables*, along with the shell script given in Appendix B.

2.3.2. The *netfilter/iptables* Package. The *netfilter/iptables* package is a software solution that integrates with the Linux kernel and allows for a variety of advanced networking capabilities. The *netfilter* portion of the package is manifested as a set of 'hooks' within the Linux kernel that permit kernel modules to register callback functions within the network stack. These registered callback functions are then called each time a packet traverses the respective hook(s) within the network stack [19]. In short, by adding these hooks and allowing the calling of functions within the network stack as traffic traverses them, *netfilter* provides a powerful means of manipulating network traffic via the Linux kernel.

The *iptables* portion of the *netfilter/iptables* package is a user-space application that allows users under a *netfilter*-installed Linux kernel to control the behavior of network traffic as it traverses the hooks described previously. Specifically, *iptables* allows a Linux administrator to define and modify rules associated with the filtering and translation of network packets as they traverse the *netfilter* hooks. Under the HarDNESS

DFA, DFNs use the *iptables* solution to apply filtering rules to traffic as it flows between the protected endpoint and the network. However, *netfilter/iptables* operates only at the network and transport layers, and is not capable of controlling network traffic at the data link layer (Appendix A). This presents a problem in terms of transparent firewalling because, to make a DFN completely transparent to its associated protected endpoint and the network, the Ethernet interfaces on the DFN connected to the associated network and protected endpoint must be bridged together, a process that occurs at the data link layer. Therefore, because of this bridge, traffic between the protected endpoint and the network will not be subjected to the rules defined by *iptables*, as the kernel considers bridged Ethernet interfaces to logically be a single Ethernet interface. To enable filtering of traffic flowing across bridged interfaces, the *ebtables* package is utilized in conjunction with the *netfilter/iptables* package.

2.3.3. The *ebtables* Package. The *ebtables* software package [20] allows a Linux-equipped device with two or more Ethernet interfaces that have been bridged together to perform filtering on the traffic flowing between those bridged interfaces. Because Ethernet bridging occurs at the data link layer, this implies that *ebtables* enables the filtering of traffic at the data link layer, dealing with Ethernet frames rather than IP packets; however, *ebtables* also possesses the ability to "peek up" into the network layer and provide filtering based upon IP parameters as well.

The functionality provided by *ebtables* is required in the proof-of-concept HarDNESS DFA, as it enables DFNs to act as Ethernet bridges when placed between network endpoints and the network to which the endpoint is attached while also enabling DFNs to apply policies to the traffic traversing the bridge. Having DFNs act as Ethernet bridges rather than as IP routers allows the DFN to be transparent to both the protected endpoint and its associated network. Therefore, transparent firewalling can be performed by the DFN without it having an IP address bound to any of its interfaces.

3. ANALYSIS

3.1. PROOF-OF-CONCEPT DISTRIBUTED SYSTEM FRAMEWORK

To operate properly, the distributed network security systems examined previously in this thesis rely in some way on the hardware and software components of the network endpoints they protect, opening such systems to possible attack vectors and preventing them from being truly device and operating system agnostic, a tenet upon which most networks are designed. One primary benefit of the HarDNESS framework is that, because the nodes need not rely on network endpoints in any fashion, it may be deployed to protect any type of endpoint device connected to an Ethernet network utilizing the TCP/IP protocol suite, including legacy, special-use, and embedded systems. Additionally, the ability to create completely transparent distributed security solutions using the HarDNESS framework minimizes or completely eliminates the need to modify configuration settings on endpoint devices.

The HarDNESS framework and the proof-of-concept implementation presented in this thesis are designed with security as being of paramount importance. Design decisions such as logically or physically separating command and control (C2) traffic from that of the network being protected by the HarDNESS deployment (Section 2.1.3), protecting the integrity of all communications between the components of HarDNESS deployments (Section 2.1.3), and the use of multiple, advanced authentication and encryption schemes (Section 2.2.2.2 and Section 2.2.2.4) limit the exposure of the system to attack by malicious entities both internal and external to the protected network. In the case of truly-out-of-band C2 deployments (Section 2.1.3), the exposure of the system to attack is exceptionally limited, as no component of the system is required to have an interface with an IP address bound to it that is attached to the network being protected. Coupled with restricting physical access to the HarDNESS components, truly-out-of-band deployments are well shielded from compromise attempts. However, because implementing a truly-out-of-band HarDNESS deployment is cost prohibitive to most organizations, it is likely that semi-out-of-band deployments are more likely to occur and, unfortunately, such deployments are more easily profiled and open to attack given that

nodes and controllers must have an interface to which an IP address is bound attached to the network being protected.

The proof-of-concept HarDNESS framework is designed to be scalable in its number of nodes and controllers. The principle of keeping the amount of controller resources per node at a minimum allows a multitude of nodes to be operated by a single controller, and the ability to deploy several controllers on a network further increases the allowable size of deployments. The ability of HarDNESS nodes to utilize backup controllers and even operate autonomously during periods of controller downtime improves the reliability and robustness of the system.

The hardware platform upon which nodes in the proof-of-concept HarDNESS framework are implemented—hardware revision 3.0 of the Linksys WRT54G wireless broadband router—provides a reasonable hardware set upon which HarDNESS applications may be built. The design decision by Linksys to provide the very vast majority of networking functionality for these devices in software renders them capable of performing an endless number of traffic-related tasks, such as filtering, manipulation, auditing, and routing. With the addition of the third-party OpenWrt firmware, upon which the node components of HarDNESS are based, tremendous flexibility in the tasks performed by these devices is realized. And, although disabled in the proof-of-concept HarDNESS framework developed in association with this thesis, the WRT54G platform allows for the possible expansion of the framework into the realm of wireless networking given its support of the prolific IEEE 802.11b/g wireless networking standards.

The flexibility of the OpenWrt firmware expands the applicability of the HarDNESS framework beyond its providing a foundation for distributed security applications. Indeed, any application requiring reasonable resources and a point-of-presence at network endpoints may be developed using the HarDNESS framework. A customizable web interface provides an easily-implemented front-end for these applications, and the ability to quickly install, remove, and update software packages on nodes via the use of the *ipkg* software package manager further simplifies the configuration of nodes for customized applications. Because OpenWrt and HarDNESS rely exclusively on open source software, components of the HarDNESS framework may easily be modified as needed. It is hoped that the flexibility provided by the HarDNESS

framework will see its expansion into networks other than those based on TCP/IP, such as those utilized by process control systems.

3.2. PROOF-OF-CONCEPT DISTRIBUTED FIREWALL APPLICATION

Deployments of the proof-of-concept HarDNESS Distributed Firewall Application (HarDNESS DFA), built as an application of the HarDNESS framework, provide the ability to perform the equivalent of conventional packet filtering firewalling between a network and any of the endpoint devices on that network where the deployments have a point of presence in the form of a distributed firewall node (DFN). In deployments where a DFN separates each endpoint device on a network from the network itself, the ability to subject all traffic flowing between a network and its endpoints to a centrally managed firewall policy exists, whereas under conventional firewalling all traffic flowing between endpoints on the same network goes unchecked by the firewall, as such traffic need not traverse the firewall to arrive at its destination. By applying firewall policies to all network traffic, a variety of threats originating from both 'inside' and 'outside' the network can be addressed.

By using a combination of software packages, the proof-of-concept HarDNESS DFA performs its firewalling services transparently, that is, without the knowledge of either the network or the endpoints being protected by the DFNs. This transparency prevents the HarDNESS DFA from being intrusive to the users and processes of network endpoints, assists in maintaining the security of the distributed system, and simplifies deployment of the distributed firewall system because no configuration changes to either network devices or the network endpoints are required.

Unlike similar distributed firewall systems that have been proposed and implemented, the basis of the HarDNESS DFA upon the HarDNESS framework implies that DFNs are completely independent of the resources of the network endpoints. Assuming that physical access to the DFNs is restricted and that physical access to a network by its endpoints is only via a DFN, the HarDNESS DFA ensures that bypassing of its firewall mechanisms is difficult and also provides advanced access control to the network itself. For example, unauthenticated endpoints can be given access only to an

authentication server within the network; upon successful authentication, the firewall rules enforced by the DFN can be adapted according to the business function of the authenticated endpoint and/or its user. Such a configuration could also prevent the 'sniffing' of network traffic, as the DFN could simply drop any network traffic not destined for its endpoint. However, as is the case where the interaction of a firewall with its protected device(s) is strictly via the network, the separation of DFNs from network endpoints prevents them from taking into consideration the intent of an endpoint when applying policies to the traffic related to the endpoint. This is unfortunate, as such knowledge can influence how a policy should be applied to traffic. Such is the case, for example, in the File Transfer Protocol (FTP), where the client and server utilize a standard control port to negotiate a random port on which data transfers will occur; knowledge of the random port to be used for data transfer would be advantageous, as that port could be temporarily opened to allow transfers to occur. Additionally, as is a universal problem in firewalling, separation of DFNs from the network endpoints renders them unable to determine the contents of encrypted traffic payloads that may traverse them.

4. SUMMARY AND CONCLUSIONS

Ethernet-based networks utilizing the TCP/IP protocol suite are ubiquitous within contemporary society. Businesses, government agencies, educational institutions, and other such organizations of all sizes and fields rely heavily upon these networks to ensure mission continuity. Increasingly, such networks are also deployed in applications where their compromise could result in serious financial, environment, and/or political harm. As organizations began connecting their networks to the Internet, the need to secure their networks against attack from external malicious entities became apparent and is today crucial in maintaining positive control over network devices and the data they process. A prominent solution to this problem manifested itself in the form of the conventional firewall, a device that sits at the border between two or more networks with different levels of trust and filters traffic flowing between those networks according to a defined policy.

As networks have grown increasingly complex and as attacks against them have followed suit, the conventional firewall concept has failed to maintain adequate pace so as to adequately thwart would-be malicious entities. Steven M. Bellovin of AT&T Labs made a promising proposal for a "distributed firewall system" that addresses many of the problems associated with conventional firewalls by maintaining central control of the firewall policies but distributing the job of enforcing the policies across the entire collection of the endpoint devices within a network. Following the release of two prominent implementations of distributed firewalls, the concept diminished within the network security research and development communities, despite its many promising benefits in shielding network devices from attack from both internal and external malicious entities.

This thesis examines some of the shortcomings associated with conventional firewalling and the two existing implementations of distributed firewall systems mentioned. To solve many of the problems associated with existing distributed firewall systems, the Hardened Distributed Network Security System (HarDNESS) framework and an application of that framework in the form of a distributed firewall system, the HarDNESS Distributed Firewall Application (HarDNESS DFA), are proposed and details

of the systems are given. Proof-of-concept implementations of the HarDNESS framework and the HarDNESS DFA have been developed as part of the research associated with this thesis, and the concepts necessary for obtaining a high-level understanding of these systems given a general background in computer networking are presented. Finally, an analysis of the benefits and shortcomings of the HarDNESS framework and HarDNESS DFA systems is provided.

In conclusion, the concept of a distributed network endpoint security system warrants further research and possible development, as a flexible solution utilizing this concept may provide astonishing means of addressing threats perpetrated against arguably the most vulnerable components of networks—their endpoint devices and ultimately the people utilizing those devices. A number of promising results have already been revealed via the proof-of-concept HarDNESS framework and HarDNESS DFA systems developed as part of the research portions associated with this thesis, and continued development will likely result in viable distributed systems for securing a variety of critical-function networks.

4.1. FUTURE DIRECTIONS

Given the short development schedule under which the proof-of-concept HarDNESS framework and HarDNESS DFA systems were constructed, future developments on the research associated with this thesis should begin with fully implementing these proof-of-concept systems, followed by a significant body of testing performed on these completed variants. Although hardware revision 3.0 of the Linksys WRT54G provides an adequate node hardware platform, further research must be performed to ensure that the best-suited node hardware platform is identified, taking into account the possibility of a custom-developed hardware platform. Additionally, the ability to integrate wireless networking into the HarDNESS framework should be taken into account, as networking trends have demonstrated that wireless networking will remain a pervasive means of network access.

Deploying medium- and large-scale testbeds of both the semi- and truly-out-of-band C2 variants of these systems should then be performed to ensure that scalability has

been adequately addressed in the design of the proof-of-concept systems, as well as to test the reasonable hypothesis that multiple controllers may be deployed without interference such that a large number of nodes may receive control services and that controller failover measures implemented on nodes do not result in significant problems for the systems.

Upon achieving adequate maturity in the HarDNESS framework and HarDNESS DFA systems, additional security capabilities, such as the advanced cryptographic verification functionalities proposed by Bellovin in [7], may then be added so as to further secure both the distributed system and the network endpoints it is intended to protect. Lastly, mature solutions should be subjected to thorough red teaming and attack analysis exercises to ensure their design and implementation is solid enough for production and deployment *en masse*.

APPENDIX A.
THE LAYERS OF THE OSI REFERENCE MODEL

The International Standards Organization (ISO) Open Systems Interconnection (OSI) Basic Reference Model consists of seven layers, described as follows:

Layer		Basic Function	Data Unit
Number	**Name**		
7	Application	Application interface to network	Data
6	Presentation	Data format handling.	
5	Session	Manage host-to-host connection sessions.	
4	Transport	Provide end-to-end connections and reliability.	Segments
3	Network	Addressing and routing.	Packets
2	Data Link	Physical addressing and logical link control.	Frames
1	Physical	Specify physical media characteristics.	Bits

Applications running on a host interact with the application layer, which is used to accept data from applications for delivery via the network. The presentation layer manipulates the format of the application data as needed, and applies a header to the data payload. The session layer then either establishes a session with the receiving host or routes the data received from the presentation layer through an established session as appropriate, appending its own header. The transport layer ensures that an end-to-end connection is established, dealing with any necessary data reliability issues, and appends an appropriate header based upon the transport protocol utilized. The network layer applies its own header and is responsible for determining the route through which the data and associated headers is to be sent. The data link layer applies a header and sends the data and determines how and when to send the data and associated headers onto the network. The physical layer provides the link by which the data and its associated headers are actually sent onto the network.

The receiving host then receives the data and associated headers off of its physical layer. As the data and headers are passed up the layers, the header for each layer is stripped off and used to determine how to process the data.

APPENDIX B.
TRANSPARENT FIREWALL SETUP SCRIPT

```
#!/bin/sh
#
# Filename: transparentFirewall.sh
# Author:   William D. Atkins <atkins.william@gmail.com>
# Date:     11 February 2007
# Purpose:  This script defines the ebtables and iptables rules
#           necessary to enable the "transparent firewall" capability
#           of a Linksys WRT54G wireless router running a distribution
#           of the OpenWRT firmware, provided that the user-space
#           ebtables and iptables utilities are installed and that the
#           kernel on the router has been patched to include ebtables
#           Layer 2 and Layer 3 frame/packet control and advanced
#           iptables traffic matching.

# List all ebtables kernel modules that must be loaded.
EBTABLES_MODULES='ebtables ebtable_broute ebtable_filter ebtable_nat
ebt_ip ebt_vlan'

# We are expecting the "network" side of the transparent firewall to be
# connected to the Ethernet port labeled as "WAN" on the Linksys
# WRT54G.  Because we want the firewall to be transparent, we do not
# assign an IP address to this interface, as no communication should be
# destined for the node on this interface.
NETWORK_IFC='vlan1'
NETWORK_IFC_IP_ADDRESS='0.0.0.0'
NETWORK_IFC_NETMASK=''

# We are expecting the "host" side of the transparent firewall to be
# connected to one of the four "LAN" Ethernet ports on the Linksys
# WRT54G.  Because we want the firewall to be transparent, we do not
# assign an IP address to this interface, as no communication should be
# destined for the node on this interface.  Note that with the
# "default" configuration of OpenWRT, the four LAN ports on the WRT54G
# are treated as a four-port switch; however, the capability to treat
# each of these four ports as separate virtual interfaces exists via
# the use of VLAN tagging.  Therefore, if desired, one of the four LAN
# ports could be specified to be connected only to the host, and
# another of the LAN ports could be connected to a completely out-of-
# band control network intended strictly for communication between
# nodes and the controller.
HOST_IFC='vlan0'
HOST_IFC_IP_ADDRESS='0.0.0.0'
HOST_IFC_NETMASK=''

# We will bridge together the "WAN" port and the four "LAN" ports, in
# essence creating a five-port Ethernet switch out of the WRT54G.  We
# will assign an IP address to this bridge interface, whether
# statically or by means of DHCP, so that management communication
# between the node and the controller can take place via an OpenVPN
# encrypted tunnel.  We deactivate the spanning tree protocol for the
# bridge, as it creates unnecessary overhead.
BRIDGE_IFC='br0'
BRIDGE_IFC_IP_ADDRESS='172.16.0.100'
BRIDGE_IFC_NETMASK='255.255.0.0'
BRIDGE_STP='off'
```

```
# Load the ebtables kernel modules.
for MODULE in $EBTABLES_MODULES; do
  insmod $MODULE
done

# Destroy all custom iptables chains and flush all rules.
for TABLE in filter nat mangle; do
  iptables -t $TABLE -F
  iptables -t $TABLE -X
done

# Destroy all custom ebtables chains and flush all rules.
for TABLE in filter nat broute; do
  ebtables -t $TABLE -F
  ebtables -t $TABLE -X
done

## AT THIS POINT NO TRAFFIC WILL TRAVERSE THE BRIDGE WHEN IT
## IS BROUGHT UP BECAUSE NO IPTABLES/EBTABLES FORWARD RULES
## HAVE BEEN ESTABLISHED, AND THE DEFAULT POLICY ON THESE
## CHAINS IS TO 'DROP' FRAMES/PACKETS THAT DO NOT MATCH ANY
## DEFINED RULES.

# Configure the "network-side" interface.
ifconfig $NETWORK_IFC inet $NETWORK_IFC_IP_ADDRESS netmask
$NETWORK_IFC_NETMASK up

# Configure the "host-side" interface
ifconfig $HOST_IFC inet $HOST_IFC_IP_ADDRESS netmask $HOST_IFC_NETMASK
up

# Create and configure the bridge interface.
brctl addbr $BRIDGE_IFC
brctl stp $BRIDGE_IFC $BRIDGE_STP
brctl addif $BRIDGE_IFC $NETWORK_IFC
brctl addif $BRIDGE_IFC $HOST_IFC
ifconfig $BRIDGE_IFC inet $BRIDGE_IFC_IP_ADDRESS netmask
$BRIDGE_IFC_NETMASK up

## FOR EXPERIMENTAL PURPOSES, WE WILL CONFIGURE THE TRANSPARENT
## FIREWALL RULES SUCH THAT ALL TRAFFIC WILL BE FORWARDED BETWEEN THE
## INTERFACES OF THE BRIDGE, ESSENTIALLY MAKING THE BRIDGE AN ETHERNET
## SWITCH.  RULES THAT RESTRICT THE TYPE OF TRAFFIC THAT ARE ALLOWED TO
## FLOW BETWEEN THE NETWORK AND THE HOST SHOULD BE DEFINED IN THE
## IPTABLES 'FILTER' TABLE, 'FORWARD' CHAIN.

# Rules for the ebtables broute table, BROUTING chain.
#NONE - default policy is to ACCEPT.

# Rules for the ebtables nat table, PREROUTING chain.
#NONE - default policy is to ACCEPT.

# Rules for the iptables mangle table, PREROUTING chain.
#NONE - default policy is to ACCEPT.

# Rules for the ebtables filter table, FORWARD chain.
# THIS IS THE TABLE/CHAIN WHERE RULES TO RESTRICT WHAT FRAMES (LAYER 2)
```

```
# ARE ALLOWED TO FLOW BETWEEN THE HOST AND THE NETWORK AND VICE-VERSA
# ARE TO BE DEFINED.  AFTER DEFINING RULES TO ALLOW DIFFERENT TYPES OF
# FRAMES, BE SURE TO ADD A 'CATCH-ALL' RULE WHOSE ACTION IS TO DROP ANY
# FRAMES THAT DID NOT MATCH THE PREVIOUSLY-DEFINED RULES.  NOTE THAT
# EBTABLES ALLOWS FOR 'PEEKING' IN TO LAYER 3 DATA ENCAPSULATED IN
# FRAMES.  ALSO NOTE THAT ANY FRAME TRANSPORTING A LAYER 3 PACKET THAT
# IS DROPPED WILL NEVER REACH THE IPTABLES FILTER TABLE, FORWARD CHAIN.
ebtables -t filter -A FORWARD -j ACCEPT

# Rules for the iptables mangle table, FORWARD chain.
#NONE - default policy is to ACCEPT.

# Rules for the iptables filter table, FORWARD chain.
# THIS IS THE TABLE/CHAIN WHERE RULES TO RESTRICT WHAT PACKETS ARE
# ALLOWED TO FLOW BETWEEN THE HOST AND THE NETWORK AND VICE-VERSA
# ARE TO BE DEFINED.  AFTER DEFINING RULES TO ALLOW DIFFERENT TYPES
# OF PACKETS, BE SURE TO ADD A 'CATCH-ALL' RULE WHOSE ACTION IS TO
# DROP ANY PACKETS THAT DID NOT MATCH THE PREVIOUSLY-DEFINED RULES.
iptables -t filter -A FORWARD -j ACCEPT

# Rules for the ebtables nat table, POSTROUTING chain.
#NONE - default policy is to ACCEPT

# Rules for the iptables mangle table, POSTROUTING chain.
#NONE - default policy is to ACCEPT

# Rules for the iptables nat table, POSTROUTING chain.
#NONE - default policy is to ACCEPT.

## THE RULES UP TO THIS POINT ALLOW ALL TRAFFIC TO BE FORWARDED
## ACROSS THE BRIDGE INTERFACES.  THE RULES BELOW DEAL WITH
## TRAFFIC ORIGINATING FROM OR DESTINED TO THE NODE ITSELF, NOT
## TRAFFIC FORWARDED BY THE NODE.

# Rules for the ebtables filter table, INPUT chain.
ebtables -t filter -A INPUT -j ACCEPT

# Rules for the iptables mangle table, INPUT chain.
#NONE - default policy is to accept.

# Rules for the iptables filter table, INPUT chain.
iptables -t filter -A INPUT -j ACCEPT

# Rules for the iptables mangle table, OUTPUT chain.
#NONE - default policy is to ACCEPT.

# Rules for the iptables nat table, OUTPUT chain.
#NONE - default policy is to accept.

# Rules for the iptables filter table, OUTPUT chain.
iptables -t filter -A OUTPUT -j ACCEPT

# Rules for the ebtables nat table, OUTPUT chain.
#NONE - default policy is to accept.

# Rules for the ebtables filter table, OUTPUT chain.
ebtables -t filter -A OUTPUT -j ACCEPT
```

BIBLIOGRAPHY

[1] http://www.crime-research.org/analytics/501/. Cyberterrorism: fear factor, March 2007.

[2] W. D. Atkins, "The Morris Worm: An Overview and Analysis," Unpublished, Midterm Paper for Computer Engineering 349, *Trustworthy, Survivable Computer Networks*, University of Missouri–Rolla, Fall 2004.

[3] E. D. Zwicky, S. Cooper, and D. B. Chapman, *Building Internet Firewalls, Second Edition*, O'Reilly & Associates, Inc., pp. 102–107, 2000.

[4] http://www.cisco.com/web/about/ac123/ac147/ac174/ac200/about_cisco_ipj_archive_article09186a00800c85ae.html. Firewalls and Internet Security, March 2007.

[5] T. W. Shinder, "Firewall Concepts" in *Best Damn Firewall Book Period*, Syngress Publishing, pp. 53–71, 2003.

[6] L. A. Gordon, M. P. Loeb, W. Lucyshyn, R. Richardson, *2005 CSI/FBI Computer Crime and Security Survey*, p. 15. Summary Available: http://www.fbi.gov/page2/july05/cyber072505.htm.

[7] S. M. Bellovin, "Distributed firewalls," *;login Magazine*, pp. 37–39, Nov. 1999. Available: http://www.cs.columbia.edu/~smb/papers/distfw.html.

[8] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith, "Implementing a Distributed Firewall," *Proceedings of the 7th ACM Conference on Computer and Communications Security*, pp. 190–199, Athens, Greece, 2000. Available: http://www1.cs.columbia.edu/~angelos/Papers/df.pdf.

[9] C. Payne and T. Markham, "Architecture and applications for a distributed embedded firewall," *Proceedings of the 17th Annual Computer Security Applications Conference*, pp. 329–336, 2001.

[10] http://www.3com.com/other/pdfs/products/en_US/101155.pdf. 3Com Embedded Firewall Solution, March 2007.

[11] http://www.thegreenbow.com/fwd.html. TheGreenBow Distributed Firewall. March, 2007.

[12] http://www.yoggie.com/. Yoggie – Laptop Security Firewall Appliance Solution. March, 2007.

[13] http://en.wikipedia.org/wiki/Wireless_LAN. Wireless LAN, March 2007.

[14] http://en.wikipedia.org/wiki/Linksys. Linksys, March, 2007.

[15] http://openwrt.org. OpenWrt, March, 2007.

[16] http://openvpn.net/. OpenVPN – An Open Source VPN Solution by James Yonan, March, 2007.

[17] http://www.zeroflux.org/cgi-bin/cvstrac.cgi/knock/wiki. knock - Wiki index, March, 2007.

[18] http://www.openssh.com/. OpenSSH, March, 2007.

[19] http://www.netfilter.org/. Netfilter/iptables project homepage, March, 2007.

[20] http://ebtables.sourceforge.net/. Ebtables, March, 2007.

VITA

William Dee Atkins was born June 24, 1982 to Dennis and Rhonda Atkins of Hooker, Oklahoma. He attended the Oklahoma School of Science and Mathematics his junior and senior years of high school and graduated in May 2001. He then attended the University of Missouri–Rolla where he received a Bachelors of Science summa cum laude in Computer Engineering with minors in Computer Science and Mathematics in May 2005. He continued his education at the University of Missouri–Rolla and in May 2007 received a Masters of Science in Computer Engineering with emphasis in computer, network, and information security.

While pursuing his Master's degree, Atkins served as a graduate research assistant to Dr. Ann Miller, Cynthia Tang Missouri Distinguished Professor of Electrical and Computer Engineering, and was a member of the UMR Trustworthy Systems Laboratory research team. Atkins also was a member of the UMR Node of the Sandia National Laboratories Center for Cyber Defenders program, completing two internships in the program at Albuquerque, New Mexico the summers of 2005 and 2006 and a co-op in the fall of 2006. Atkins continued at Sandia National Laboratories as a year-round graduate student intern through the summer of 2007 and is pending hire as a full-time employee working in critical infrastructure protection.