BUILDING LINUX FIREWALL

by

Mr. Satit Phermsawang

A Final Report of the Three-Credit Course
CE 6998 Project

Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Science
in Computer and Engineering Management
Assumption University

March 2003

BUILDING LINUX FIREWALL

by
Mr. Satit Phermsawang

A Final Report of the Three-Credit Course
CE 6998 Project

Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Science
in Computer and Engineering Management
Assumption University

March 2003

Project Title	Building Linux Firewall
Name	Mr. Satit Phermsawang
Project Advisor	Dr. Chamnong Jungthirapanich
Academic Year	March 2003

The Graduate School of Assumption University has approved this final report of the three-credit course, CE 6998 PROJECT, submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer and Engineering Management.

Approval Committee:

[illegible]
(Dr. Chamnong Jungthirapanich)
Dean and Advisor

[illegible]
(Prof.Dr. Srisakdi Charmonman)
Chairman

[illegible]
(Assoc.Prof. Somchai Thayarnyong)
MUA Representative

March 2003

ABSTRACT

This project will give you the basic knowledge and concepts of the functions of a firewall. There are many types of firewalls and types of attacks to be studied. This project will also touch upon topics of why a firewall is needed and what it will do to benefit the user. This project has been well thought out and will teach you how to setup a firewall without having any prior knowledge as to how a firewall works. A basic, safe, and easy to use system has been set up. By using Phayoune Firewall you will see a Thai security Linux distribution in action. The types and different varieties of attacks are always changing, thus there is no perfect firewall to keep up with the changing times. It is important to examine your firewall and update it often. With the popularity of the Internet and more and more people coming on-line, it is important to create these walls of security, to protect your data and information.

Hackers are out there, and getting smarter by the day, thus it is important to take action now, and protect yourself from this new type of cyber crime.

ACKNOWLEDGEMENTS

The writer's indebted to the following people and organizations. Without them, this project would not been possible.

The writer wishes to express sincere gratitude to his advisor, Dr. Chamnong Jungthirapanich.whose patient assistance, guidance, and constant encouragement has led him from the research inception to the research completion

The writer would like to thank Aron Andrus, Pramonsri and the staff at the Department of Information Technology, Wimol Business Administration Collage for their help in construction of Phayoune Firewall Prototype.

Special appreciation is due to The writer's family for their fervent and continuous encouragement. Above all, he forever grateful to his boss whose willingness to invest in my future has enabled him to achieve his educational goal.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

1.1 Overview

Security is increasingly important for companies and individuals alike. The Internet has provided them with a powerful tool to distribute information about them and obtain information from others, but it has also exposed them to dangers that they have previously been exempt from. Computer crime, information theft, and malicious damage are all potential dangers

An unauthorized and unscrupulous person who gains access to a computer system may guess system passwords or exploit the bugs and idiosyncratic behavior of certain programs to obtain a working account on that machine. Once they are able to log in to the machine, they may have access to information that may be damaging, such as commercially sensitive information like marketing plans, new project details, or customer information databases. Damaging or modifying this type of data can cause severe setbacks to the company.

The safest way to avoid such widespread damage is to prevent unauthorized people from gaining network access to the machine. This is where firewalls come in. Constructing secure firewalls are an art. It involves a good understanding of technology, but equally important, it requires an understanding of the philosophy behind firewall designs. I won't cover everything you need to know in this project; I strongly recommend you do some additional research before trusting any particular firewall design, including any I had presented here.

1.2 Objective

This Project Design manual will outline the critical elements for implementing an Internet firewall in your organization, hopefully in a way that allows you to make by yourself the Firewall solutions that best suit your needs. The solutions are intended to get a secure and cheap firewall. This project has been well thought out and will teach you how to setup a firewall without having any prior knowledge as to how a firewall works.

1.3 Scope

(1) To understand the Firewall basic and types. These describe the firewall basics and types on literature review which included some Internet securities.

(2) To understand and measure the network securities. You will know how to search and use network security tools, such as IPTABLES from www.netfilter.org, Phayoune Firewall from www.phayoune.org, etc.

(3) To understand and setup some kinds of Free Phayoune Linux operation system. On the last project I will focus on how to building your Linux firewall that suits your network I will address the Linux-specific technique by using Phayoune Secure Linux, one of Thai Linux Distributions, released by GPL, configuration that should serve as a useful starting point in your own step by step configuration, but as with all security-related matters, trust no one. Double check the design; make sure you understand it, and then modify it to suit your requirements.

II. LITERATURE REVIEW

2.1 What Is an Internet Firewall?

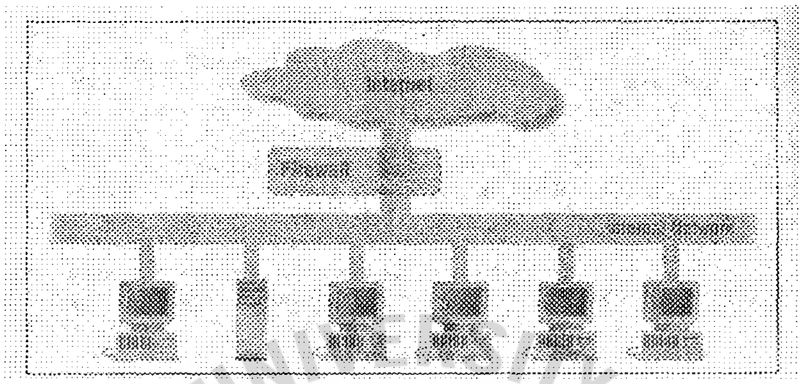


Figure 2.1. Firewall Model.

A system designed to prevent unauthorized access to or from a private network. Firewalls can be implemented in both hardware and software, or a combination of both. Firewalls are frequently used to prevent unauthorized Internet users from accessing private networks connected to the Internet, especially intranets. All messages entering or leaving the intranet pass through the firewall, which examines each message and blocks those that do not meet the specified security criteria.

In networking, a firewall could be described as a specially designed device that controls the spreading of a network threat. The most commonly talked about source of network threats is the Internet. The Internet is the home of many unknown people that we cannot trust. There are hackers on the Internet that may want to do our networks harm.

We can use a firewall to impede an untrusted person from doing damage to our networks.

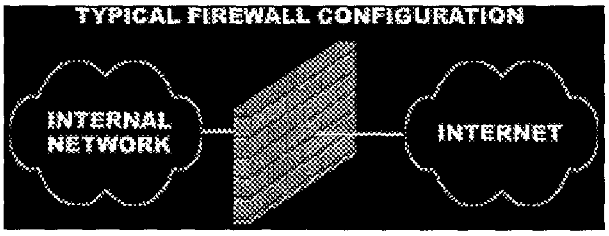


Figure 2.2. Typical Firewall Configuration.

A more textbook definition of a computer firewall is that it is a method or device that regulates the level of trust between two or more networks. A firewall can consist of software, hardware or a combination of both. A firewall can protect your network from the Internet as well as regulate the traffic between networks within the same company.

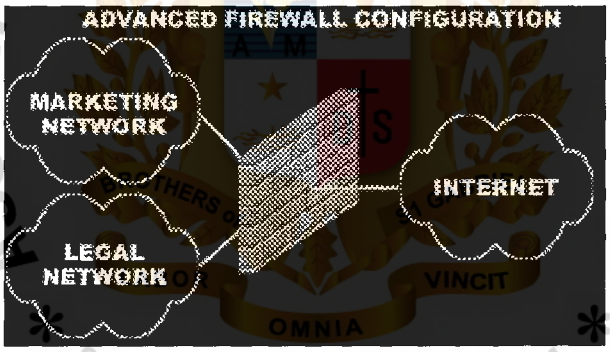


Figure 2.3. Advanced Firewall Configuration.

For instance, a firewall can allow the legal department's network to have access to the marketing file server but the marketing department can be refused access to legals. In this example the firewall is positioned between the marketing and legal networks so that all communication must pass through the firewall. The firewall is then able to ensure that only authorized packets are allowed

2.2 Type of Firewalls

The level of protection firewalls provides and the ways they offer that protection vary widely. However, broadly speaking, most commercially available firewalls fall into one of four categories

(1) Packet Filtering Firewall

(2) Circuit-level gateway

(3) Application-level gateway

(4) Proxy Server or Stateful inspection firewall

Few firewalls belong in only one of these categories, and fewer still exactly match the definition I will offer for any one category. Nevertheless, these definitions reflect the key capabilities that differentiate one firewall from another.

Packet-filtering Firewall

A packet-filtering firewall is a router or computer running software that has been configured to screen incoming and outgoing packets. A packet-filtering firewall accepts or denies packets based on information contained in the packets' TCP and IP headers. For example, most packet-filtering firewalls can accept or deny a packet based on the packet's full association, which consists of the following:

Source address

Destination address

Application or protocol

Source port number

Destination port number

All routers (even those that are not configured to filter packets) routinely check the full association to determine where to send the packets they receive. However, a packet-filtering firewall goes one step further: Before forwarding a packet, the firewall compares the full association against a table containing rules that dictate whether the firewall should deny or permit packets to pass.

A packet-filtering firewall scans these rules until it finds one that agrees with the information in a packet's full association. If the firewall encounters a packet that does not meet one of the rules, the firewall will apply the default rule. A default rule should be explicitly defined in the firewall's table and, for strict security, should instruct the firewall to drop a packet that meets none of the other rules.

Rules to Live By. You can define packet-filtering rules that indicate which packets should be accepted and which packets should be denied. For example, you could configure rules that instructed the firewall to drop packets from specific untrusted servers (generally called hosts on the Internet), which you would identify in the table by their IP addresses. You could also create a rule that permitted only incoming e-mail messages traveling to your mail server and another rule that blocked incoming e-mail messages from an untrusted host that had flooded your network with several gigabytes of data in the past.

In addition, you can configure a packet-filtering firewall to screen packets based on TCP and User Datagrams Protocol (UDP) port numbers. Configuring a firewall in this way enables you to implement a rule that tells the firewall to permit particular types of connections (such as Telnet and FTP connections) only if they are traveling to appropriate trusted servers (such as the Telnet and FTP server, respectively). However, the success of such a rule depends on a TCP/IP network convention: Servers (and clients) generally run particular TCP/IP applications over particular ports (often referred to as well-known ports), but servers are not required to use these ports.

Low Cost for Relatively Low Protection? The primary advantage of using a packet-filtering firewall is that it provides some measure of protection for relatively low cost and causes little to no delay in network performance. If you already have an IP router with packet-filtering capabilities, setting up a packet-filtering firewall will cost no more than the time it takes to create packet-filtering rules. Most IP routers, including those manufactured by Novell, Cisco Systems, and Bay Networks, can filter incoming and outgoing packets.

Although the cost of a packet-filtering firewall is attractive, this firewall alone is often not secure enough to keep out hackers with more than a passing interest in your network. Configuring packet-filtering rules can be difficult, and even if you manage to create effective rules, a packet-filtering firewall has inherent limitations. For example, suppose you created a rule that instructed the firewall to drop incoming packets with unknown source addresses. This rule would make it more difficult--but not impossible--for a hacker to access at least some trusted servers with IP addresses: The hacker could simply substitute the actual source address on a malicious packet with the source address of a trusted client.

Layer Upon Layer. In addition, a packet-filtering firewall primarily operates only at the network layer of the Open Systems Interconnection (OSI) model. The OSI model, which was developed by the International Standards Organization (ISO), identifies the seven layers at which computers communicate, ranging from the physical media over which they communicate to the applications they use to communicate.

All firewalls rely on information generated by protocols that function at various layers of the OSI model. Knowing the OSI layer at which a firewall operates is one of the keys to understanding different types of firewalls. Generally speaking, the higher the OSI layer at which a firewall filters packets, the greater the level of protection the firewall provides.

Because a packet-filtering firewall generally checks information only in IP packet headers, sneaking packets through this type of firewall is relatively easy: A hacker simply creates packet headers that satisfy the firewall's rules for permitting packets. Beyond that, a packet-filtering firewall cannot detect the contents of a packet.

Circuit-level Gateway

A circuit-level gateway monitors TCP handshaking between packets from trusted clients or servers to untrusted hosts and vice versa to determine whether a requested session is legitimate. To filter packets in this way, a circuit-level gateway relies on data contained in the packet headers for the Internet's TCP session-layer protocol. Because a circuit-level gateway filters packets at the session layer of the OSI model, this gateway operates two layers higher than a packet-filtering firewall does.

Monitoring Handshaking--Circuitously. To determine whether a requested session is legitimate, a circuit-level gateway uses a process similar to the following: A trusted client requests a service, and the gateway accepts this request, assuming that the client meets basic filtering criteria (such as whether DNS can locate the client's IP address and associated name).

Next, acting on behalf of the client, the gateway opens a connection to the requested untrusted host and then closely monitors the TCP handshaking that follows. This handshaking involves an exchange of TCP packets that are flagged SYN (synchronize) or ACK (acknowledge). These packet types are legitimate only at certain points during the session. See the SYNDefender white paper for a more detailed description of the SYN/ACK process.

A circuit-level gateway determines that a requested session is legitimate only if the SYN flags, ACK flags, and sequence numbers involved in the TCP handshaking between the trusted client and the untrusted host are logical.

Pipe Proxies. After a circuit-level gateway determines that the trusted client and the untrusted host are authorized to participate in a TCP session and verifies the legitimacy of this session, the gateway establishes a connection. From this point on, the circuit-level gateway simply copies and forwards packets back and forth without further filtering them.

The gateway maintains a table of established connections, allowing data to pass when session information matches an entry in the table. When the session is completed, the gateway removes the associated entry in the table and closes the circuit this session used.

A circuit-level gateway relies on special applications to perform copy and forward services. These applications are sometimes called pipe (or generic) proxies because they establish a virtual circuit, or pipe, between two networks and then allow packets (generated by one or more types of TCP/IP applications) to pass through this pipe.

Seldom Standalone. Because pipe proxies generally support several TCP/IP services, a circuit-level gateway can extend the number of services supported by an application-level gateway, which relies on application-specific proxies. In fact, most circuit-level gateways are not stand-alone products but instead are packaged with application-level gateways.

Proxy Server Protection. A circuit-level gateway provides one other important security function: It is a proxy server. Although the term proxy server suggests a server that runs proxies (which is true of a circuit-level gateway), the term actually means something different. A proxy server is a firewall that uses a process called address translation to map all of your internal IP addresses to one "safe" IP address. This address is associated with the firewall from which all outgoing packets originate.

As a result, on a network with a circuit-level gateway, all outgoing packets appear to have originated from that gateway, preventing direct contact between the trusted network and the untrusted network. That is, a circuit-level gateway's IP address is the only active IP address and the only IP address that the untrusted network is aware of. Thus, a circuit-level gateway and other proxy servers protect trusted networks from spoofing attacks.

Circumventing Circuits. A circuit-level gateway does have one inherently vulnerable characteristic, however: Once a circuit-level gateway establishes a connection, any application can run across that connection because a circuit-level gateway filters packets only at the session layer of the OSI model. In other words, a circuit-level gateway cannot examine the application-level content of the packets it relays between a trusted network and an untrusted network.

Because a circuit-level gateway does not filter individual packets but blindly relays packets back and forth across established connections, a hacker on an untrusted network could possibly slip malicious packets past the gateway. The hacker could then deal directly with an internal server, such as a WWW server, which may not be as carefully monitored or configured as the firewall itself.

As long as the initial TCP packets exchanged between the trusted WWW server and the untrusted host met the handshaking criteria, the gateway would establish a connection and copy and forward subsequent packets--regardless of their content. To filter the application-level content of individual packets generated by particular services, you need an application-level gateway.

Application-level Gateway

Like a circuit-level gateway, an application-level gateway intercepts incoming and outgoing packets, runs proxies that copy and forward information across the gateway, and functions as a proxy server, preventing any direct connection between a trusted server or client and an untrusted host. However, the proxies that an application-level gateway runs differ in two important ways from the pipe proxies that a circuit-level gateway uses:

The proxies are application specific.

The proxies can filter packets at the application layer of the OSI model.

Application-specific Proxies. Unlike pipe proxies, application-specific proxies accept only packets generated by services they are designed to copy, forward, and filter. For example, only a Telnet proxy can copy, forward, and filter Telnet traffic. If a network relies only on an application-level gateway, incoming and outgoing packets cannot access services for which there is not a proxy. For example, if an application-level gateway ran FTP and Telnet proxies, only packets generated by these services could pass through the firewall. All other services would be blocked.

Application-level Filtering. Unlike a circuit-level gateway, an application-level gateway runs proxies that examine and filter individual packets, rather than simply copying them and blindly forwarding them across the gateway. Application-specific proxies check each packet that passes through the gateway, verifying the contents of the packet up through the application layer (which is the highest layer) of the OSI model. These proxies can filter particular kinds of commands or information in the application protocols the proxies are designed to copy, forward, and filter.

Application gateways can also restrict specific actions from being performed. For example, the gateway could be configured to prevent users from performing the FTP put command. This command lets users write to the FTP server. Prohibiting this action can prevent serious damage of the information stored on the server.

Transparency--Ah, There's the Rub! An application-level gateway is one of the most secure firewalls available, but some vendors (usually those that market stateful inspection firewalls) and users claim that the security an application-level gateway offers has a drawback--lack of transparency. Ideally, an application-level gateway would be as transparent as it is secure. Users on the trusted network would not notice that they were accessing Internet services through a firewall. In reality, however, users often experience delays or must perform multiple logins before they are connected to the Internet or an intranet via an application-level gateway.

Although most vendors claim that application-level gateways are transparent, many vendors recommend that you configure the gateway to require user authentication before users access an untrusted network, a process that foils true transparency.

Some firewall vendors that market products as application-level gateways have tried to overcome the transparency problem. For example, one particular application gateway uses a version of the SOCKS protocol (rather than application-specific proxies) to route TCP/IP services. SOCKS is a proposed Internet Engineering Task Force (IETF) standard that provides transparent authentication services for clients requesting connections to devices through firewalls. However, a SOCKS server is not transparent to network administrators: You must modify the applications running on each client that will use the firewall.

Also, although SOCKS includes other security features (such as private-key and public-key encryption), it does not filter individual packets. Therefore, the products that rely on SOCKS might fall justifiably into the realm of circuit-level gateways rather than application-level gateways.

Proxy Server or Stateful Inspection Firewall

A server that sits between a client application, such as a Web browser, and a real server. It intercepts all requests to the real server to see if it can fulfill the requests itself. If not, it forwards the request to the real server. Proxy servers have two main purposes:

Improve Performance: Proxy servers can dramatically improve performance for groups of users. This is because it saves the results of all requests for a certain amount of time. Consider the case where both user X and user Y access the World Wide Web through a proxy server. First user X requests a certain Web page, which we'll call Page 1. Sometime later, user Y requests the same page. Instead of forwarding the request to the Web server where Page 1 resides, which can be a time-consuming operation, the proxy server simply returns the Page 1 that it already fetched for user X. Since the proxy server is often on the same network as the user, this is a much faster operation. Real proxy servers support hundreds or thousands of users. The major online services such as Compuserve and America Online, for example, employ an array of proxy servers.

Filter Requests: Proxy servers can also be used to filter requests. For example, a company might use a proxy server to prevent its employees from accessing a specific set of Web sites.

A stateful inspection firewall combines aspects of a packet-filtering firewall, a circuit-level gateway, and an application-level gateway. Like a packet-filtering firewall, a stateful inspection firewall operates at the network layer of the OSI model, filtering all incoming and outgoing packets based on source and destination IP addresses and port numbers.

A stateful inspection firewall also functions as a circuit-level gateway, determining whether the packets in a session are appropriate. For example, a stateful inspection firewall verifies that SYN and ACK flags and sequence numbers are logical. Finally, a stateful inspection firewall mimics an application-level gateway: The firewall evaluates the contents of each packet up through the application layer and ensures that these contents match the rules in your company's network security policy.

Better Performance, Same Level of Security? Like an application-level gateway, a stateful inspection firewall can be configured to drop packets that contain specific commands. For example, you could configure a stateful inspection firewall to drop FTP packets containing a Put or Get command.

Unlike an application-level gateway, however, a stateful inspection firewall does not break the client-server model to analyze application-layer data. An application-level gateway requires two connections: one connection between the trusted client and the gateway and another connection between the gateway and the untrusted host. The gateway then relays information between the two connections. Although some people insist that this configuration ensures the highest degree of security, other people argue that this configuration slows performance unnecessarily.

A stateful inspection firewall, on the other hand, does not require two connections, allowing a direct connection between a trusted client and an untrusted host. To provide a secure connection, a stateful inspection firewall intercepts and examines each packet up through the application layer of the OSI model.

Rather than relying on application-specific proxies (and thus limiting users to the services for which you are running a proxy), a stateful inspection firewall relies on algorithms to recognize and process application-layer data. These algorithms compare packets against known bit patterns of authorized packets and are theoretically able to filter packets more efficiently than application-specific proxies.

Because a stateful inspection firewall allows a direct connection between a trusted client and an untrusted host, some people believe this firewall is less secure than an application-level gateway. However, other people argue that using a direct connection makes a stateful inspection firewall perform better than an application-level gateway at no cost to security.

What's Out There? A stateful inspection firewall is a popular solution for securing Internet and intranet connections because this firewall is transparent to users, scrutinizes data at the highest OSI layer, and does not require you to modify clients or run a separate proxy for each service that runs over the firewall. In fact, Check Point Software Technologies, Ltd.'s FireWall-1, which is one of the most popular commercial firewalls, is a stateful inspection firewall. Credited with coining the term stateful inspection, Check Point began selling FireWall-1 in 1993 and now owns 44 percent of the firewall market.

Don't be Careless. Stateful inspection firewalls are among the most secure firewalls available today and "fooling them can be a lot of work," according to Jon McCown, a network security analyst for the U.S. National Computer Security Agency (NCSA). Nevertheless, stateful inspection firewalls, like all firewalls are not 100 percent effective. So why bother implementing a firewall at all?. In practice, many firewalls use two or more of these techniques in concert. A firewall is considered a first line of defense in protecting private information. For greater security, data can be encrypted.

2.3 Why Firewall?

The Internet is a worldwide-connected network. New it can provide many services for our everyday activities such as communication, downloading program, information searching, and international business development.

It is scarcely possible to enter a bookstore, read a magazine or a newspaper, or listen to news broadcast without seeing or hearing something about the Internet in some guise. It's become so popular that no advertisement is complete without a reference to a web page. While non-technical publications are obsessed with the Internet, the technical publications have moved on and are obsessed with security. It's a logical progression; once the first excitement of having a superhighway in your neighborhood wears off, you're bound to notice that not only does it let you travel, it lets a very large number of strangers show up where you are, and not all of them are people your would have invited.

Bother views are true: The Internet is a marvelous technological advance that provides access to information, and the ability to publish information, in revolutionary ways. But it's also a major danger that provides the ability to pollute and destroy information in revolutionary ways. This project is about one way to balance the advantages and the risks-to take part in the Internet while still protecting yourself.

2.4 What Are You Trying to Protect?

A firewall is basically a protective device. If you are building a firewall, the first thing you need to worry about is what you're trying to protect. When you connect to the Internet, you're putting three things at risk:

(1) Your data: the information you keep on the computers.

(2) Your resources: the computers themselves.

(3) Your reputation.

Your Data

Your data has three separate characteristics that need to be protected:

(a)	Secrecy:	You might not want other people to know it.
(b)	Integrity:	You probably don't want other people to change it.
(c)	Availability:	You almost certainly want to be able to use it yourself.

People tend to focus on the risks associated with secrecy, and it's true that those are usually large risks. Many organizations have some of their most important secrets-the designs for their products, financial records, or student records-on their computers. On the other hand, you may find that at your site it is relatively easy to separate the machines containing this kind of highly secret data from the machines that connect to the Internet. (Or you may not; you can't do Internet electronic commerce without having information about orders and money pass through Internet-accessible machines.)

Suppose that you can separate your data in this way, and that none of the information that is Internet accessible is secret. In that case, why should you worry about security? Because secrecy isn't the only thing you're trying to protect. You still need to worry about integrity and availability. After all, if your data isn't secret, and if you don't mind it being changed, and if you don't care whether or not anybody can get to it, why are you wasting disk space on it?

Even if your data isn't particularly secret, you'll suffer the consequences if it's destroyed or modified. Some of these consequences have readily calculable costs: if you lose data, you'll have to pay to have it reconstructed; if you were planning to sell that data in some form, you'll have lost sales regardless of whether the data is something you sell directly, the designs from which you build things, or the code for a software product. Intangible costs are also associated with any security incident. The most serious is the loss of confidence (user confidence, customer confidence, investor confidence, staff confidence, student confidence, and public confidence) in your systems and data and, consequently, a loss of confidence in your organization.

Your Resources

Even if you have data you don't care about-if you enjoy reinstalling your operating system every week because it exercises the disks, or something like that-if other people are going to use your computers, you probably would like to benefit from this use in some way. Most people want to use their own computers, or they want to charge other people for using them. Even people who give away computer time and disk spaces usually expect to get good publicity and thanks for it; they aren't going to get it from intruders. You spend good time and money on your computing resources and it is your right to determine how they are used.

Intruders often argue that they are using only excess resources; as a consequence, their intrusions don't cost their victims anything. There are two problems with this argument.

First, it's impossible for an intruder to determine successfully what resources are excess and use only those. It may look as if your system has oceans of empty disk space and hours of unused computing time; in fact, though, you might be just about to start computing animation sequences that is going to use every bit and every microsecond. An intruder can't give back your resources when you want them. (Along the same lines, I don't ordinarily use my car between midnight and 6 A.M., but that doesn't mean I'm willing to lend it to you without being asked. What if I have an early morning flight the next day, or what if I'm called out to deal with an emergency?)

Second, it's your right to use your resources the way you want to, even if you merely feel some sort of Zen joy at the sight of empty disk space, or if you like the way the blink lights look when nothing's happening on your computer. Computing resources are not natural resources that belong by right to the world at large, nor are they limited resources that are wasted or destroyed if they're not used.

Your Reputation

An intruder appears on the Internet with your identity. Anything he or she does appears to come from you. What are the consequences?

Most of the time, the consequences are simply that other sites-or law enforcement agencies-start calling you to ask why you're trying to break into their systems. (This isn't as rare an occurrence as it may seem. One site got serious about security when its system administration staff added a line item to their time cards for conversations with the FBI about break-in attempts originating from their site.)

Sometimes, such impostors cost you a lot more than lost time. An intruder who actively dislikes you, or simply takes pleasure in making life difficult for strangers, may change your web site, send electronic mail, or post news messages that purport to come from you. Generally, people who choose to do this aim for maximum hatefulness, rather than believability, but even if only a few people believe these messages, the cleanup can be long and humiliating. Anything even remotely believable can do permanent damage to your reputation.

A few years ago, an impostor posing as a Texas A&M professor sent out hate email containing racist comments to thousands of recipients. The impostor was never found, and the professor is still dealing with the repercussions of the forged messages. In another case, a student at Dartmouth sent out email over the signature of a professor late one night during exam period. Claiming a family emergency, the forged email canceled the next day's exam, and only a few students showed up.

It's possible to forge electronic mail or news without gaining access to a site, but it's much easier to show that a message is a forgery if it's generated from outside the forged site. The messages coming from an intruder who has gained access to your site will look exactly like yours because they are yours. An intruder will also have access to all kinds of details that an external forger won't. For example, an intruder has all of your mailing lists available and knows exactly who you send mail to.

Currently, attacks that replace web sites are very popular; one list shows more than 160 successful attacks where sites were replaced, in 18 countries, in a single month. Many of those attacks simply replaced the sites with boasting by the attackers, but a significant portion of them was directed at the content of the sites. A site that should have touted Al Gore's suitability for the U.S. presidency was replaced by a similar anti-Gore site, for instance; political movements in Peru, Mexico, and China put up slogans; and there's no need to feel safe merely because your site concerns frivolity, as pop stars, Pro Wrestling, and the Boston Lyric Opera all suffered as well.

Even if an intruder does use your identity, a break-in at your site isn't good for your reputation. It shakes people's confidence in your organization. In addition, most intruders will attempt to go from your machines to others, which is going to make their next victims think of your site as a platform for computer criminals. Many intruders will also use compromised sites as distribution sites for pirated software, pornography, and/or other stolen information, which is not going to endear you to many folks either. Whether or not it's your fault, having your name linked to other intrusions, software piracy, and pornography is hard to recover from.

2.5 What Are You Trying to Protect Against?

What's out there to worry about? What types of attacks are you likely to face on the Internet, and what types of attackers are likely to be carrying them out? And what about simple accidents or stupidity? In the sections that follow, we touch on these topics, but we don't go into any technical detail; later chapters describe different kinds of attacks in some detail and explain how firewalls can help protect against them.

(1) Types of Attacks

There are many types of attacks on systems, and many ways of categorizing these attacks. In this section, we break attacks down into three basic categories: intrusion, denial of service, and information theft.

(2) Intrusion

The most common attacks on your systems are intrusions; with intrusions, people are actually able to use your computers. Most attackers want to use your computers as if they were legitimate users.

Attackers have dozens of ways to get access. They range from social engineering attacks (you figure out the name of somebody high up in the company; you call a system administrator, claiming to be that person and claiming to need your password changed right now, so that you can get important work done), to simple guesswork (you try account names and password combinations until one works), to intricate ways to get in without needing to know an account name and a password.

As we described in this project, the firewalls will help you to prevent intrusions in a number of ways. Ideally, they block all ways to get into a system without knowing an account name and password. Properly configured, they reduce the number of accounts accessible from the outside that are therefore vulnerable to guesswork or social engineering. Most people configure their firewalls to use one-time passwords that prevent guessing attacks. Even if you don't use these passwords, Authentication and Auditing Services, a firewall will give you a controlled place to log attempts to get into your system, and, in this way, they help you detect guessing attacks.

(3) Denial of service

A denial of service attack is one that's aimed entirely at preventing you from using your own computers.

In late 1994, victim Josh Quitters and MichelleSpatula ,Quitter-Slatalla, were the targets of an "electronic mail bomb". Apparently in retaliation for an article on the cracker community they'd published in Wired magazine, someone broke into IBM, Sprint, and the writers' network provider, and modified programs so their email and telephone service was disrupted. A flood of email messages so overwhelmed their network service that other messages couldn't get through; eventually, their Internet connection was shut down entirely. Their phone service also fell victim to the intruders, who reprogrammed the service so those callers were routed to an out-of-state number where they heard an obscene recording. Although some cases of electronic sabotage involve the actual destruction or shutting down of equipment or data, more often they follow the pattern of flooding seen in the Quittner-Slatalla case or in the case of the 1988 Morris Internet worm. An intruder so floods a system or network-with messages, processes, or network requests-that no real work can be done. The system or network spends all its time responding to messages and requests, and can't satisfy any of them.

While flooding is the simplest and most common way to carry out a denial of service attack, a cleverer attacker can also disable services, reroute them, or replace them. For example, the phone attack in the Quittner-

Slatalla case denied phone service by rerouting their phone calls elsewhere; it's possible to mount the same kind of attack against Internet services. It's close to impossible to avoid all denial of service attacks. Sometimes it's a "heads, I win; tails, you lose" situation for attackers. For example, many sites set accounts up to become unusable after a certain number of failed login attempts. This prevents attackers from simply trying passwords until they find the right one. On the other hand, it gives the attackers an easy way to mount a denial of service attack: they can lock any user's account simply by trying to log in a few times.

Most often, the risk of denial of service attacks is unavoidable. If you accept things from the external universe-electronic mail, telephone calls, or packages-it is possible to get flooded. The notorious college prank of ordering a pizza or two from every pizzeria in town to be delivered to your least favorite person is a form of denial of service; it's hard to do much else while arguing with 42 pizza deliverers. In the electronic world, denial of service is as likely to happen by accident as on purpose (have you ever had a persistent fax machine tring to fax something to your voice line?). The most important thing is to set up services so that if one of them is flooded, the rest of your site keeps functioning while you find and fix the problem.

Flooding attacks are considered unsporting by many attackers, because they aren't very difficult to carry out. For most attackers, they're also pointless, because they don't provide the attacker with the information or the ability to use your computers (the payoff for most other attacks). Intentional flooding attacks are usually the work of people who are angry at your site in particular, and at most sites such people are quite rare. With the right tools and cooperation, it's fairly easy to trace flood packets back to their source, but that might not help you figure out who is behind the attacks. The attacks almost always come from machines that have they been broken into; only a really stupid attacker generates an easily traced flood of packets from their own machine. Sometimes flooding attacks are carried out by remote control. Attackers install remotely controlled flooding software on systems that they break into over the course of many weeks or months. This software lies dormant and undiscovered until some later time, when they trigger many of these remotely controlled installations simultaneously to bombard their victims with massive floods of traffic from many different directions at once. This was the method behind the highly publicized denial of service attacks on Yahoo!, CNN, and other high-profile Internet sites early in the year 2000. On the other hand, some denial of service attacks is easier for attackers, and these are relatively popular. Attacks that involve sending small amounts of data that cause machines to reboot or hang are very popular with the same sort of people who like to set off fire alarms in dormitories in the middle of the night, for much the same reason; with a small investment, you can massively annoy a very large number of people who are unlikely to be able to find you afterwards. The good news is that most of these attacks are avoidable; a well-designed firewall will usually not be susceptible to them itself, and will usually prevent them from reaching internal machines that are vulnerable to them.

(4) Information theft

Some types of attacks allow an attacker to get data without ever having to directly use your computers. Usually these attacks exploit Internet services that are intended to give out information, inducing the services to give out more information than was intended, or to give it out to the wrong people. Many Internet services are designed for use on local area networks, and don't have the type or degree of security that would allow them to be used safely across the Internet.

Information theft doesn't need to be active or particularly technical. People who want to find out personal information could simply call you and ask (perhaps pretending to be somebody who had a right to know): this is an active information theft. Or they could tap your telephone: this is a passive information theft. Similarly, people who want to gather electronic information could actively query for it (perhaps pretending to be a machine or a user with valid access) or could passively tap the network and wait for it to flow by.

Most people who steal information try to get access to your computers; they're looking for usernames and passwords. Fortunately for them, and unfortunately for everybody else, that's the easiest kind of information to get when tapping a network. Use name and password information occurs quite predictably at the beginning of many network interactions, and such information can often be reused in the same form.

How would you proceed if you want to find out how somebody answers her telephone? Installing a tap would be an easy and reliable way to get that information, and a tap at a central point in the telephone system would yield the telephone greetings of hundreds or thousands of people in a short period of time.

On the other hand, what if you want to know how somebody spells his or her last name, or what the names and ages of his or her children are? In this case, a telephone tap is a slow and unreliable way to get that information. A telephone tap at a central point in the system will probably yield that information about some people, and it will certainly yield some secret information you could use in interesting ways, but the information is going to be buried among the conversations of hundreds of people setting up lunch dates and chatting about the weather.

Similarly, network taps, which are usually called snifters, are very effective at finding password information but are rarely used by attackers to gather other kinds of information. Getting more specific information about a site requires both extreme dedication and patience, or the knowledge that the information you want will reliably pass through a given place at a given time. For example, if you know that somebody calls the bank to transfer money between his or her checking and savings accounts at 2 P.M. every other Friday, it's worth tapping that phone call to find out the person's access codes and account numbers. However, it's probably not worth tapping somebody else's phone, on the off chance that they too will do such a transfer, because most people don't transfer money over the phone at all.

Network sniffing is much easier than tapping a telephone line. Historically, the connectors used to hook a computer to an Ethernet network were known as network taps (that's why the term tapping isn't used for spying on a network), and the connectors behave like taps too. In most networks, computers can see traffic that is intended for other hosts. Traffic that crosses the Internet may cross any number of local area networks, any one of which can be a point of compromise. Network service providers and public-access systems are very popular targets for intrusions; snuffers placed there can be extremely successful because so much traffic passes through these networks.

There are several types of protection against information theft. A properly configured firewall will protect you against people who are trying to get more information than you intended to give. Once you've decided to give information out across the Internet, however, it's very difficult to protect against that information's reaching an unintended audience, either through misauthentication (somebody claiming to be authorized, when he or she isn't) or through sniffing (somebody simply reading information as it crosses a correctly authorized channel). For that matter, once you have given the information to somebody, you have no way to prevent that person from distributing it to other people. Although these risks are outside of the protection a firewall can give (because they occur once information has intentionally been allowed to go outside your network).

2.6 Types of Attackers

This section very briefly describes the types of attackers who are out there on the Internet. There are many ways to categorize these attackers; we can't really do justice to the many variants of attackers we've seen over the years, and any quick summary of this kind necessarily presents a rather stereotyped view. Nevertheless, this summary may be useful in distinguishing the main categories of attackers.

All attackers share certain characteristics. They don't want to be caught, so they try to conceal themselves, their identity and real geographic location. If they gain access to your system, they will certainly attempt to preserve that access, if possible, by building in extra ways to get access (and they hope you won't notice these access routes even if you find the attackers themselves). Most of them have some contact with other people who have the same kinds of interests ("the underground" is not hard to find), and most will share the information they get from attacking your system. A secondary group of attackers may not be as benign.

(1) Joyrides

Joyriders are bored people looking for amusement. They break in because they think you might have interesting data, or because it would be amusing to use your computers, or because they have nothing better to do. They might be out to learn about the kind of computer you have or about the data you have. They're curious but not actively malicious; however, they often damage the system through ignorance or in trying to cover their tracks. Joy riders are particularly attracted to well-known sites and uncommon computers.

(2) Vandals

Vandals are out to do damage, either because they get their kicks from destroying things, or because they don't like you. When one gets to you, you'll know it. Vandals are a big problem if you're somebody that the Internet underground might think of as The Enemy (for example, the phone company or the movement) or if you tend to annoy people who have computers and time (for example, you're a university with failing students, or a computer company with annoyed customers, or you have an aggressively commercial presence on the Internet). You can also become a target simply by being large and visible; if you put a big wall up in certain neighborhoods, people will put graffiti on it no matter how they feel about you.

Fortunately, vandals are fairly rare. People don't like them, even people in the underground who have nothing against breaking into computers in general. Vandals also tend to inspire people to go to great lengths to find them and stop them. Unlike more mundane intruders, vandals have short but splashy careers. Most of them also go for straightforward destruction, which is unpleasant but is relatively easily detected and repaired. In most circumstances, deleting your data, or even ruining your computer equipment, is not the worst thing somebody could do to you, but it is what vandals do. (Actually, introducing subtle but significant changes in programs or financial data would be much harder to detect and fix.)

Unfortunately, it's close to impossible to stop a determined vandal; somebody with a true vendetta against your site is going to get you, sooner or later. Certain attacks are attractive to vandals but not to other types of attackers. For example, denial of service attacks are not attractive to joyriders; while joy riders are around in your system. they are just as interested as you are in having your computers up, running, and available to the Internet.

(3) Scorekeepers

Many intruders are engaging in an updated version of an ancient tradition. They're gaining bragging rights, based on the number and types of systems they've broken into.

Like joyrides and vandals, scorekeepers may prefer sites of particular interest. Breaking into something well known, well defended, or otherwise especially cool is usually worth more points to them. However, they'll also attack anything they can get at; they're going for quantity as well as quality. They don't have to want anything you've got or care in the least about the characteristics of your site. They may or may not do damage on the way through. They'll certainly gather information and keep it for later use (perhaps using it to barter with other attackers). They'll probably try to leave themselves ways to get back in later. And, if at all possible, they'll use your machines as a platform to attack others.

These people are the ones you discover long after they've broken in to your system. You may find out slowly, because something's odd about your machine. Or you'll find out when another site or a law enforcement agency calls up because your system is being used to attack other places. Or you'll find out when somebody sends you a copy of your own private data, which they've found on a cracked system on the other side of the world.

Many scorekeepers are what are known as script kiddies-attackers who are not themselves technically expert but are using programs or scripts written by other people and following instructions about how to use them. Although they do tend to be young, they're called "kiddies" mostly out of contempt aimed at them by more experienced intruders. Even though these attackers are not innovators, they still pose a real threat to sites that don't keep rigorously up to date. Information spreads very rapidly in the underground, and the script kiddies are extremely numerous. Once a script exists, somebody is almost guaranteed to attack your site with it. These days, some scorekeepers aren't even counting machines they've broken into but are keeping score on crashed machines. On the other hand, having a machine crash is generally less destructive than having it broken into; on the other hand, if a particular attack gets into the hands of the script kiddies, and thousands of people use it to crash your machine, it's not funny any more.

(4) Spies (industrial and otherwise)

Most people who break into computers do so for the same reason people climb mountains-because they're there. While these people are not above theft, they usually steal things that are directly convertible into money or further access (e.g., credit card, telephone, or network access information). If they find secrets they think they can sell, they may try to do so, but that's not their main business. As far as anybody knows, serious computer-based espionage is much rarer, outside of traditional espionage circles. (That is, if you're a professional spy, other professional spies are probably watching you and your computers.) Espionage is much more difficult to detect than run-of-the-mill break-ins, however. Information theft need not leave any traces at all, and even intrusions are relatively rarely detected immediately. Somebody who breaks in, copies data, and leaves without disturbing anything is quite likely to get away with it at most sites.

In practical terms, most organizations can't prevent spies from succeeding. The precautions that governments take to protect sensitive information on computers are complex, expensive, and cumbersome; therefore, they are used on only the most critical resources. These precautions include electromagnetic shielding, careful access controls, and absolutely no connections to unsecured networks.

What can you do to protect against attackers of this kind? You can ensure that your Internet connection isn't the easiest way for a spy to gather information. You don't want some kid to break into your computers and find something that immediately appears to be worth trying to sell to spies; you don't want your competitors to be trivially able to get to your data; and you do want to make it expensive and risky to spy on you. Some people say it's unreasonable to protect data from network access when somebody could get it easily by coming to your site physically. We don't agree; physical access is generally more expensive and more risky for an attacker than network access.

2.7 Who Do You Trust?

Much of security is about trust; who do you trust to do what? The world doesn't work unless you trust some people to do some things, and security people sometimes seem to take an overly suspicious attitude, trusting nobody. Why shouldn't you trust your users, or rich, famous software vendors?

We all know that in day-to-day life there are various kinds of trust. There are people you would lend a thousand dollars but not tell a secret to; people you would ask to baby-sit but not lend a book to; people you love dearly but don't let them touch the good china because they break things. The same is true in a computer context. Trusting your employees not to steal data and sell it is not the same thing as trusting them not to give it out by accident. Trusting your software vendor not to sell your software designed to destroy your computer is not at all the same thing as trusting the same vendor not to let other people destroy your computer.

You don't need to believe that the world is full of horrible, malicious people who are trying to attack you. You do need to believe that the world has some horrible, malicious people who are trying to attack you, and is full of really nice people who don't always pay attention to what they're doing.

When you give somebody private information, you're trusting them two ways. First, you trust them not to do anything bad with it; second, you trust them not to let anybody else steal it. Most of the time, most people worry about the first problem. In the computer context, you need to explicitly remember to think about the second problem. If you give somebody a credit card number on paper, you have a good idea what procedures are used to protect it, and you can influence him or her. If carbon sheets are used to make copies, you can destroy them. If you give somebody a credit card electronically, you trust not only their honesty but also their skill at computer security. It's perfectly reasonable to worry about the latter even if the former is impeccable.

If the people who use your computers and who write your software are all trustworthy computer security experts, great; but if they're not, decide whether you trust their expertise separately from deciding whether you trust their honesty.

2.8 Packet-Filtering Concepts

The term firewall has a number of different meanings depending on the mechanisms used to implement the firewall, the level of the TCP/IP protocol stack the firewall is operating on, and the network and routing architectures used. Three of the most common meanings refer to a packet-filtering firewall; an application gateway, also called a screened-host firewall; and an application-level circuit gateway, also called a proxy firewall.

A packet-filtering firewall is normally implemented within the operating system and operates at the IP network and transport protocol layers. It protects the system by making routing decisions after filtering packets based on information in the IP packet header. An application gateway, or screened-host firewall, is implemented at the network architecture and system configuration levels. Network traffic is never passed through the application gateway machine. External access is allowed only to the gateway machine. Internal access is allowed only to the gateway machine. Local users must log in to the gateway machine and access the Internet from there. Additionally, the gateway machine may be protected by packet-filtering firewalls on both its external and internal interfaces.

A proxy firewall is usually implemented as separate applications for each service being proxies. Each proxy application appears to be the server to the client program, and appears to be the client to the real server. Special client programs, or specially configured client programs, connect to the proxy server instead of a remote server.The proxy establishes the connection to the remote server on the client application's behalf, after substituting the client's source address with its own. Proxy applications can ensure data integrity-that is, that data appropriate to the service is being exchanged-filter against viruses, and enforce high-level, detailed access control policies.

A packet-filtering firewall consists of a list of acceptance and denial rules. These rules explicitly define which packets will and will not be allowed through the network interface. The firewall rules use the packet header fields to decide whether to route a packet through to its destination, to silently throw the packet away, or to block the packet and return an error condition to the sending machine. These rules are based on the specific network interface card and host IP address, the network layer's source and destination IP addresses, the transport layer's TCP and UDP service ports, TCP connection flags, the network layer's ICMP message types, and whether the packet is incoming or outgoing.

Using a hybrid of the TCP/IP reference model, a packet-filtering firewall functions at the network and transport layers, as shown in Figure 2.4.

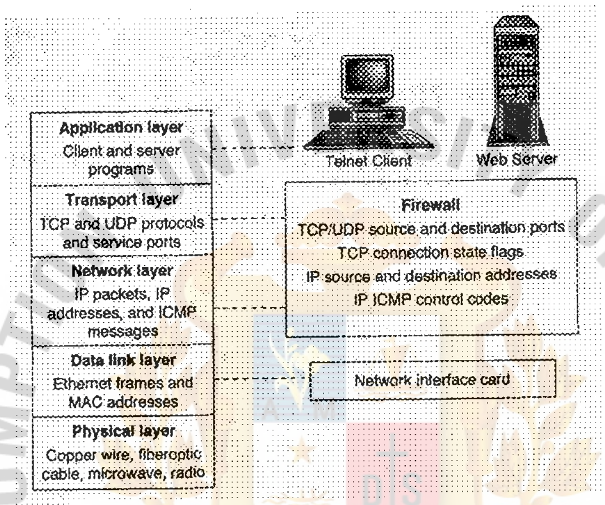

Application layer
Client and server programs
Transport layer
TCP and UDP protocols and service ports
Network layer
IP packets, IP addresses, and ICMP messages
Data link layer
Ethernet frames and MAC addresses
Physical layer
Copper wire, fiberoptic cable, microwave, radio
Telnet Client
Web Server
Firewall
TCP/UDP source and destination ports
TCP connection state flags
IP source and destination addresses
IP ICMP control codes
Network interface card

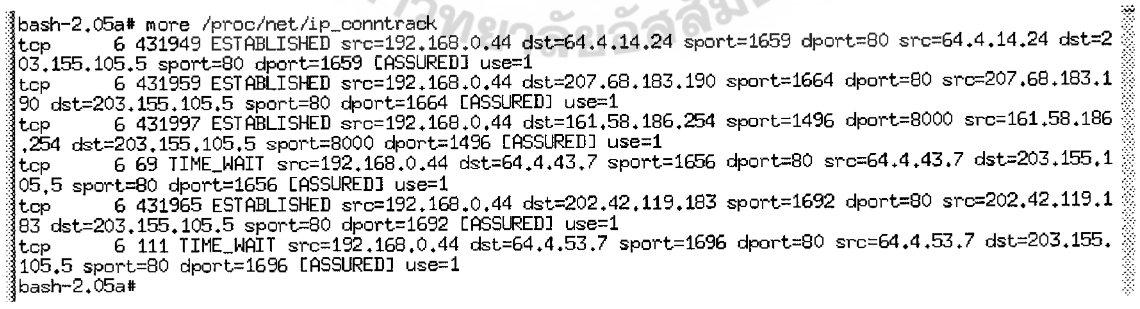
Figure 2.4. Packet-Filtering Firewall Functions at the Network and Transport Layers.

III. SYSTEM DEVELOPMENT

3.1 Netfilter or Iptables

When Linux 2.4 was released, most people focused on what it would do to help the average Linux user and talked about the USB support, firewire, PCMCIA and DRI. While these are great additions to the kernel for the majority of people, often one of the major improvements over 2.2 was overlooked, even though it applies almost as much to Joe (and of course Jane) User as it does to a hardened network engineer. This is, of course, the inclusion of the 'netfilter' system into the kernel, which provides packet filtering and other more advanced IP features. Along with 'netfilter' comes 'iptables', which is the 2.4 equivalent of ipchains, and provides a user-space interface to the filtering, Network Address Translation (NAT) and mangling modules.

We're going to look at building 2.4 with support for netfilter and iptables, then building a production level router out of it. For those of you who just have one machine, and use it to connect to the Internet, then many of the same rules apply. The Internet is one giant, generally unrestricted, network which any reasonable person would have reservations about putting any sort of machine on, never mind their own Linux system.



```
bash-2.05a# more /proc/net/ip_conntrack
tcp      6 431949 ESTABLISHED src=192.168.0.44 dst=64.4.14.24 sport=1659 dport=80 src=64.4.14.24 dst=2
03.155.105.5 sport=80 dport=1659 [ASSURED] use=1
tcp      6 431959 ESTABLISHED src=192.168.0.44 dst=207.68.183.190 sport=1664 dport=80 src=207.68.183.1
90 dst=203.155.105.5 sport=80 dport=1664 [ASSURED] use=1
tcp      6 431997 ESTABLISHED src=192.168.0.44 dst=161.58.186.254 sport=1496 dport=8000 src=161.58.186
.254 dst=203.155.105.5 sport=8000 dport=1496 [ASSURED] use=1
tcp      6 69 TIME_WAIT src=192.168.0.44 dst=64.4.43.7 sport=1656 dport=80 src=64.4.43.7 dst=203.155.1
05.5 sport=80 dport=1656 [ASSURED] use=1
tcp      6 431965 ESTABLISHED src=192.168.0.44 dst=202.42.119.183 sport=1692 dport=80 src=202.42.119.1
83 dst=203.155.105.5 sport=80 dport=1692 [ASSURED] use=1
tcp      6 111 TIME_WAIT src=192.168.0.44 dst=64.4.53.7 sport=1696 dport=80 src=64.4.53.7 dst=203.155.
105.5 sport=80 dport=1696 [ASSURED] use=1
bash-2.05a#
```

Figure 3.1. Contracts Means You Can Find out Which Connection Specific Packets Are Associated with.

(1) Netfilter or Iptables?

Often when referring to the firewalling code in 2.4, it will blindly be referred to as 'netfilter' or 'iptables', without any justification for using the specific name for it and, given that they are both very different, it's worth understanding exactly what each of them do and how we should view the organisation of the firewalling code in the kernel.

Netfilter is the system compiled into the kernel which provides hooks into the IP stack which loadable modules (iptables is one) can use to perform operations on packets. As netfilter uses modules for the filtering, you can use an ipchains module to provide exactly the same capabilities as the kernel level ipchains code in 2.2, or even the module for ipfwadm from 2.0. Netfilter is there all of the time, as long as it is compiled in, whether or not you are using any firewalling modules at all.

IPTables is split into two parts; the user-space tools and the kernel-space modules. The kernel-space modules are distributed with the main kernel, and you compile them as you would any other module, be it sound drivers, a filesystem or USB support. There is the main ip_tables module, as well as modules specifically for NAT, logging, connection tracking and so on. These modules perform the appropriate function on the packets which they get sent by netfilter, depending on the rules which they have in their rule-list, or chain.

The user-space iptables code comes in the form of a binary called 'iptables', which is distributed separately from the main kernel tree, and is used to add, remove or edit rules for the modules. This is comparable to the ipchains binary in 2.2. Often, when referring to iptables, it is assumed to mean the iptables binary, and we will continue to use such a standard here.

(2) Configure, compile, install, reboot

Ideally, you need to have a machine which is already running a 2.4 kernel, or have the knowledge to install 2.4 on a machine currently running 2.2., as the required updates to make sure 2.4 runs without problems are outside the scope of this article. This machine is going to be for mission critical routing, so the use of the latest bleeding edge kernel is not really necessary; all we need is something which is stable, secure and is not going to corrupt our filesystems.

Aside from all of the other options which you may or may not need, there are numerous settings under 'Networking Options' which don't directly pertain to iptables, but are applicable to many features of it. Firstly, we need to select 'Network Packet Filtering', which basically enables the use of netfilter, although unless you're intending to become a netfilter developer, you won't need the debugging option. You will probably also want to enable 'IP: advances router' and 'IP: use netfilter MARK value as routing key'. We next need to compile some modules which netfilter can use, with the 'IP: Netfilter Configuration' sub-menu. Everything there needs to be selected as m, apart from the 2.2 and 2.0 support, unless you specifically need to use ipchains or ipfwadm on the machine while you learn to use iptables.

As with any kernel rebuild, make dep && make clean && make bzlilo && make modules && make modules_install, then reboot, assuming you are using lilo.

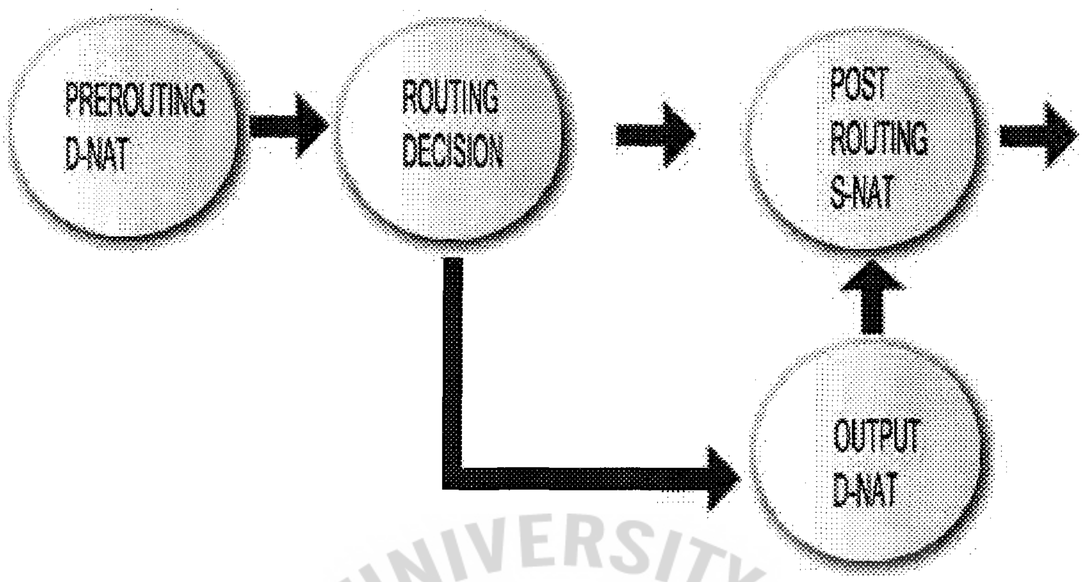


Figure 3.2. When a Packet Reaches a Junction, It Is Examined to See If Any Action Should Be Taken or If It Should Move onto the Next Chain.

(3) IPTable

Once the new kernel is up and running happily, we can go ahead and compile the userspace iptables tool. You can download this, the latest release being 1.2, from http://netfilter.kernelnotes.org/. is basically a matter of doing make; make install, as root, and everything is sorted out. You will need a configured 2.4 kernel available for iptables to compile against, so if you've not yet built 2.4, or have deleted the source code, you might want to take a few steps back and have another go.

Next we need to load the ip_tables module into the kernel using modprobe ip_tables. A lot of the other modules are loaded automatically as we use the various features, but both ip_nat_ftp and ip_conntrack_ftp need to be loaded manually, and we will look at their usage later.

(4) Filtering

As with ipchains, iptables has three lists of rules – or chains – for filtering. For those of you who are confused about moving from ipchains, they have exactly the same names, but have to be in upper case, so there is INPUT, OUTPUT and FORWARD. INPUT applies to all packets destined for the local machine, OUTPUT for packets which originated locally and FORWARD for packets which are sent to our machine, but are not actually for it.

We can, if we choose, create our own chains to organise our rules into different groups based on other rules. We create a rule with iptables -N <rule-name> and delete it with iptables –X <rule-name>. After this, they behave just like the three default chains, and we can flush them with iptables -F <rule-name> or list their rules with iptables -nL <rule-name>.

Using iptables we can perform three actions on the chains which alter their rules. We can either add, insert or delete rules, using -A, -I and -D, respectively, followed by the chain name. So, if we wanted to add another rule to the end of the INPUT chain we would use iptables -A INPUT. Not much use so far, as we need to specify which packets we want the rule to apply to. Matching source and destination IPs and ports is the most straight forward things to do. If, for example, we want to block all connections to port 23, over tcp, to a local machine we would do: iptables -A INPUT -p tcp --dport 23 -j DROP -p sets the IP protocol used, be it TCP, ICMP, UDP or one of the other more unused protocols, and --dport specifies the destination port of the packet. We can, of course, use --sport to specify a source port, but that is rarely used as connections usually use a random source port, unless they are from a specific service, such as NTP or BIND which has packets coming from a specific port.

Those who migrated from ipchains will be familiar with the difference between DENY and REJECT. However, the people who wrote iptables thought that DENY and REJECT sounded like the same thing, so there is now DROP and DENY. DROP literally drops the packet without making any effort to clean up afterwards, whereas DENY drops the packet, then returns an ICMP packet to the source of the packet to tell it that the connection was denied.

Describing source and destination IP addresses is often used to distinguish between trusted and unknown networks, and there are a number of different ways to refer to IPs and network addresses with iptables. Within the rule, we can use -s <ipaddr> and -d <ip-addr> to set the source and destination Ips which must match for the rule to be used. We can either use a normal IP, such as 10.1.2.4, a hostname, such as mail.domain.com or a network address 10.1.2.0/24 as an example. In the latter case, it will use either the common slash notation or a proper network address/netmask identifier, and an IP is of course really a /32. If we neglect to use either -s or -d it will use 0.0.0.0/0.0.0.0 which will match any packet.

Often, we want to drop all internal traffic coming in from a remote network, such as the Internet, and this can be done with a combination of the -d flag, and -i which refers to the input network device, such as ppp0 or eth0:

```
iptables -A INPUT -d 10.0.0.0/8 -i ppp0 -j DROP
```

INPUT will only understand -i, and OUTPUT -o, as neither will have a device of the opposite type, but a rule in FORWARD can use both -i and -o, as it is not unlikely that a packet will come in one interface and go out of another, depending on the routing.

Previously, we would check for the SYN flag, which is usually indicative of a packet which is going to start a new connection, in order to prevent incoming connections to a machine.

Unfortunately, this is not a particularly secure way to do it, as it is fairly straightforward to create software which starts connections with malformed packets, and even if you can't do that, there are plenty of things you can download off the Internet which will do all the hard work for you. IPTables has a far better option, in the form of connection tracking modules. Every time a new connection is created, either locally, or by routing through our machine from somewhere else, ip_conntrack catches it and stores the details, so it can use the information to see which connection specific packets belong to. Now, rather than just checking for the SYN flag, we can check to see if it matches a currently established connection, which is much neater.

There are four types of connection which can exist. NEW corresponds to packets which are being used to create new connections. This is, of course, done by checking the connection tracking list, rather than checking any packet flags, so will apply to new connections being routed through our machine. ESTABLISHED relates to packets from a known connection, and RELATED applies to packets related to a active connection, such as an ICMP reply, or via the use of ip_conntrack_ftp, active FTP sessions. Last, but by no means last, is INVALID which should be dropped and are malformed or unrecognised packets.

All this matching is done with the -m switch, and for connection tracking, or stateful matching, we use -m state followed by a --state option, then list the packet types we do, or don't, want to match.

If we, for example, wanted to drop all NEW or INVALID packets coming in ppp0 we would use:

```
iptables -A INPUT -m state --state NEW, INVALID -j DROP
```

Sickeningly easy, isn't it? We can have it match packets which do not match a specific state:

```
iptables -A INPUT -m state --state ! INVALID -j INCOMING
```

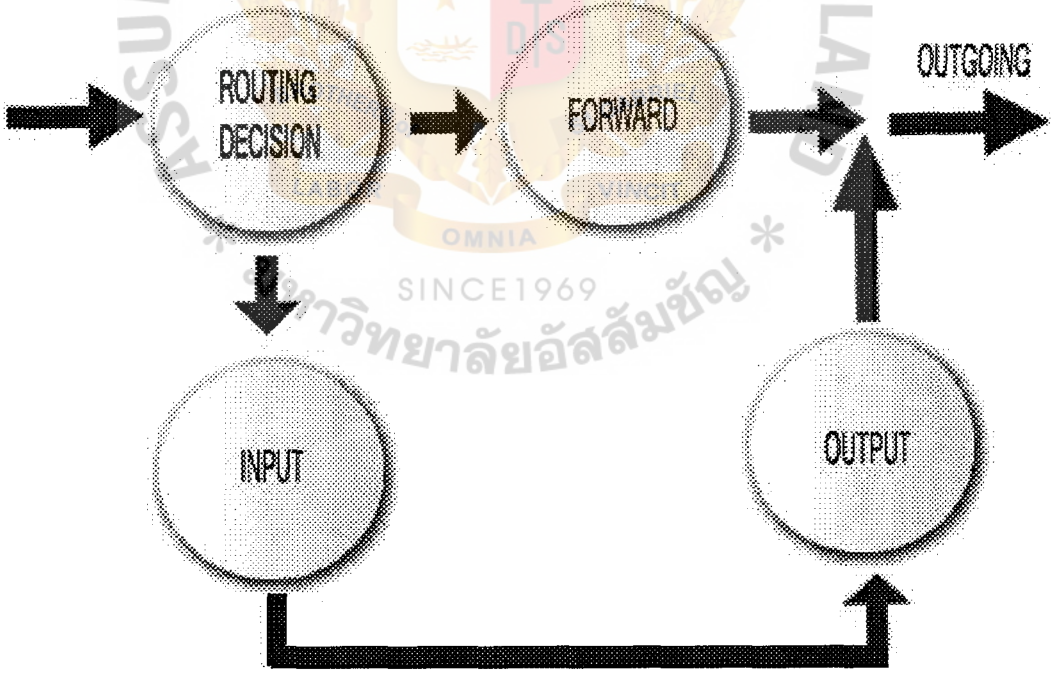


Figure 3.3. Each Rule Is Examined against the Three Chains until a Match Is Found.

Which would match all non-INVALID packets, then start to match them against rules in the INCOMING chain.

Matching specific flags of TCP packets is still important, so we can still check them using the --tcp-flags. This is slightly different, as it takes two options. Firstly, it needs a list of all flags it should check, then a list of the flags which should be set. If we wanted to perform a check for a 'SYN packet', that is, a packet with the SYN flag set, we would do;

```
iptables -A INPUT -p tcp --tcp-flags SYN, ACK, FIN SYN –j DROP
```

This translated into English says: drop all packets which have the SYN flag set, and the ACK and FIN flags not set. All other TCP flags are not checked. The above is provided as a single option -- syn, so:

```
iptables -A INPUT -p tcp --syn -j DROP
```

is exactly the same.

The -m flag can match numerous other things, such as source MAC address, which is useful on a network where you only want trusted physical machines accessing services, but the most important match is the rate limiting, which is very useful for log messages, or limiting the connections on a machine. -m limit is followed by --limit, which sets the rate limiting. If we use --limit 1/s it will allow one packet every second to match the rule. You will, however, notice that this doesn't work straight away. As default, it will allow the first five packets straight through, which is not always what we want. The --limit-burst option specifies the number of packets which can match the rule before it starts to limit the packets;

```
iptables -A INPUT -m limit --limit 5/m --limit-burst 10 --syn –j ACCEPT
```

This will allow the first ten packets to pass without any interruption, then limit to one packet every twelve seconds, or five per minute. Every time we hit a limit time, but a packet has not passed, one is added to the current burst, so if we had ten packets, then none for a whole minute, it would allow five packets to pass through before starting to limit them again. After two minutes of no packets, the burst will be back to the beginning and will allow the first ten packets through again.

This is especially useful for logging, as we don't want our logs being filled up with loads of repeated information. Unlike ipchains, logging is done with a LOG target, much like ACCEPT or DROP. However, unlike the others, even if a packet matches a defined LOG rule, it will continue to transcend the chain, so you would normally have:

```
iptables -A INPUT -i ppp0 -m state --state NEW -m limit --limit 1/m --limit-burst 0 -j LOG

iptables -A INPUT -i ppp0 -m state --state NEW -j DROP
```

in order to log, then drop, all incoming connections on ppp0.

LOG will take two optional arguments, --log-level allowing you to specify a syslogd level, such as debug, info, etc, and --logprefix, which lets you set a textual prefix to the log entry, up to twenty-nine characters, so you can easily distinguish which rule threw up the log entry.

(5) NAT

So far, we've looked at rules, chains, targets and matches, but as it is called iptables, there must be a table of some sort in there. Well, unknown to us, we've been using a table all along, although it is actually the default, so we didn't notice. 'filter' is the table used to filter packets, and contains the three chains, plus any ones created using the iptables utility. There are two other tables, nat and mangle, which also exist, and we're going to look at the nat table first, as it contains some of the most important changes to the 2.4 kernel over 2.2.

Firstly, we can list the chains in the nat table with:

```
iptables -t nat -nL
```

Which will throw up PREROUTING, POSTROUTING and OUTPUT. These three chains, much like the chains in filter, only apply to certain packets, although they are a little broader. Packets pass through the PREROUTING chain when they enter the machine, whether they are destined for the local machine or for somewhere else, before any routing decisions are taken by the kernel, so it doesn't know where they are going. OUTPUT corresponds to any packet originating from the local machine, and POSTROUTING to any packets leaving our machine, after the routing decision is taken, but did not originate locally.

We can, if we really wanted to, DROP, ACCEPT or LOG packets using these chains, but they are mainly used for Network Address Translation, or NAT, features. You might, at first, think that it does not apply to you, but IP masquerading is a type of NAT, so if you intend to share a network connection, it may be worth paying attention.

There are two varieties of NAT, source NAT or destination NAT. It doesn't take an experienced network administrator to work out that source NAT changes the source address, or port, of packets and destination NAT changes the destination information. Because of the way NAT works, source NAT, or SNAT, only works in the POSTROUTING chain, and DNAT in the PREROUTING or OUTPUT chains.

The most obvious reason to use NAT is to traffic packets from a public network, such as the Internet, onto an internal LAN, then back out again. We might want to have all SMTP connections to our router from the outside world forwarded onto our mail server on our LAN, and to do this we would use DNAT:

```
iptables -A PREROUTING -p tcp --dport 25 -i ppp0 -j DNAT -- to 10.1.1.2:25
```

And, that is it. It will track packets coming in, and going out, so the outside world does not notice anything odd is going on, even if there are no masquerading or SNAT rules for the internal machines. The only caveat is that all packets for the connection must pass through the router, so you can't traffic packets from the mail server via a different machine to the outside world, as it will not know that it should reverse the DNAT rule when something leaves the network.

SNAT is used to hide internal IPs behind public IPs, which is not quite like masquerading. We might want to have all packets coming from 10.1.1.4 to correspond to a specific external IP:

```
iptables -A POSTROUTING -s 10.1.1.4 -o ppp0 -j SNAT –to 192.168.1.2
```

Of course, 192.168.1.2 is not a public address range, but it is just an example. What you can do with SNAT depends on the IP allocation, if any, from your ISP, so you may only be able to SNAT onto a single IP, but you can have many SNAT rules for a single external IP.

IP Masquerading is a form of SNAT, except that it is more interested in the interface than the IP address:

```
iptables -A POSTROUTING -o ppp0 -j MASQUERADE
```

Will masquerade all packets going out of ppp0, just like in 2.2, but it is worth knowing what the difference is. Masqueraded connections are handled just like SNAT, until the interface goes down, at which point all connections are dropped, and you have to start again. Obviously, if you have a dynamic IP, you won't want old connections with the wrong IP address hanging around, as they won't do anything useful, but if you have a static IP then you suddenly lose the ability to resume TCP connections when you redial, as the router won't remember how it SNATed them the last time.

Problems are encountered when we SNAT onto an IP belonging to an interface which the packet will not be going out of, as SNAT only changes the packet, not the routing, in the POSTROUTING chain. This can be combated by changing the routing for specific packets, which we will cover next.

(6) MANGLE

The mangle table is a little strange, as it is used to change packet properties, which won't have a direct effect on them. As with nat, mangle has the same three chains which we can use to set packet properties, specifically marking them for later rules. Marking packets is especially useful if you want to use something outside of iptables, such as iproute2, to perform an operation on a specific packet, but it cannot match all of the options we need. If we wanted to mark all packets heading for a SMTP server with the number 1 we would do:

```
iptables -t mangle -p tcp --dport 25 -j MARK --mark 1
```

You might wonder exactly what the point of that is, but it is the first step in performing per-packet routing, as iproute2 can be used to setup routing tables based on 'fwmark', rather than the traditional destination IP as 'route' does. This is quite handy if you have a quick, but unreliable DSL connection, and a slow and stable ISDN line, but want all of your mail heading across the ISDN line.

(7) Application

So, we now know most of the theory, but what about using it in practice? If we just have a single machine connected to the Internet and don't want to allow any incoming connections, but want to masquerade our LAN behind it we just do:

```
iptables -A INPUT -m state --state NEW,INVALID -i ppp0 –j DROP
```

```
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

Not forgetting to do:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Notice that we don't need any ugly rules for active FTP, as we use ip_conntrack_ftp, which stops the incoming FTP connections being tracked as NEW. We might want to allow some ports, such as 113 and 25, for identd and SMTP mail delivery. We just do:

```
iptables -I INPUT 1 -p tcp -m multiport --dport 113,25 –j ACCEPT
```

multiport is a matching option which can be used to specify multiple ports within the same rule, either --dport or --sport.

To have this happen every time we reboot, we can use a combination of iptables-save and iptables-restore to save and restore the rules, or we can just write a bash script, or pop it on the end of /etc/rc.d/rc.local, depending on the distribution. Usually, if you're just starting out, it is best to use a script, as you can change it, rerun it, and you can be 100% sure that if you reboot the machine, it will end up how you have it now. However, this isn't much use if you've got a couple of hundred machines behind a firewall and want to run proper web and mail servers, and allow internal machines to access the outside world transparently.

The simplest way to do this is to setup a selection of 10.1.x.0/24 networks, and put different classes of machines on them, such as front end servers, back end servers and workstations, as we will want to apply rules to each class in order to secure the network. The actual internal structure of the network depends on the services used, but it is not best to plug backend servers onto a hub along with a load of web servers. The public IPs will all be allocated to the public interface on the router, so the internal machines need not care which IP they are using, let alone how anyone gets to them.

Firstly, we need to take control of the IPs we are going to use, which is nothing more than making sure the public side of the LAN, which will probably consist of little more than a router for the line, knows where to send packets destined for the IPs we have. The quickest way is to setup IP Aliases for the network interface which faces the outside world, eth0 in our case, so we might have 'eth0:mail' as the interface for the mail server's IP. We could, instead, use arp to publish the NICs MAC addresses relationship to the IP with arp -Ds 192.168.1.2 eth0 pub, where 192.168.1.2 is the public IP. Which ever method is chosen, it will have to be performed whenever the machine is rebooted, so it should be inserted within the iptables setup script.

We will want all packets going to 192.168.1.2 to head for our mail server 10.1.1.2, and anything coming from 10.1.1.2, that is new connections, to look as if they are from 192.168.1.2:

iptables -t nat -A PREROUTING -p tcp -i eth0 -d 192.168.1.2 --dport 25 -j DNAT --to 10.1.1.2:25

iptables -t nat -A POSTROUTING -s 10.1.1.2 -o eth0 -j SNAT --to 192.168.1.2

All packets coming from our workstation network should only come out of one IP, which is easily done with another SNAT rule:

iptables -t nat -A POSTROUTING -s 10.1.3.0/24 -o eth0 –j SNAT --to 192.168.1.1

We can also drop packets from the PREROUTING chain, which makes it easy to drop all incoming connections to any machine, which does not have a specific DNAT rule, and as we'reperforming operations based on interface, rather than IP, we don't need to explicitly allow 10/8 traffic from being routed through our machine.

We hit a problem with this sort of setup, as if 10.1.3.2 hits our public IP for the mail server 192.168.1.2, the router translates it to 10.1.1.2, so the mail server gets a packet to 10.1.1.2 from 10.1.3.2, which won't travel back through our router in order for the DNAT to be reversed. This is quickly and easily combated with a SNAT rule, which will make all internal connections to any of our public IPs look as if they are coming from the router, and the DNAT will reverse correctly:

iptables -t nat -A POSTROUTING -i eth1 -d 192.168.1.0/24 –j SNAT --to 10.1.1.1

Assuming our router has the internal IP of 10.1.1.1 on eth1, we can extend this further, to force all of the workstations to use a squid web cache which lives on 10.1.1.3:3128. A simple DNAT rule:

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 –s 10.1.3.0/24 -j
DNAT --to 10.1.1.3:3128
```

Squid needs a couple of options to perform correctly as a transparent cache, but those are well documented at http://www.squid-cache.org.

(8) Conclusion

By now, we should have some variety of network running with iptables, and once you've spent time working out all its eccentricities, you'll wonder how you ever got along with ipchains. Even for a single machine connected to the Internet with a 56k modem, as a large proportion of people are, the sheer simplicity of its use makes it very difficult for even the most inexperienced user not to make the effort and having a go, assuming they can get 2.4 up and running in the first place. Simply, iptables offers many, many features which ipchains is technically incapable of, and when your own network is sitting on the Internet all hours of the day and night, you're not going to pick second best.

IV. SYSTEM EVALUATION

4.1 Firewall Installation

I would first of all like to introduce to you how this system will benefit you. In the market there is plenty of limited distribution of Linux products. Most of these products have a large distribution of packages such as Red Hat, Suse, Mandrake. These systems are too large and not specific enough for you to use as a firewall. In this instance I have decided to produce a small, fast and secure Linux to become a simple and good firewall for you to setup. We call this system Phayoune Linux. It requires no hard disk and is ready to run straight from CD ROM. Configurations are all kept on floppy disks, which makes the system easy and safe to use.

(1) Get the source

There are the latest version of Phayoune Linux available for download on www.phayoune.org. There are also some basic documents for who want to learn Linux.

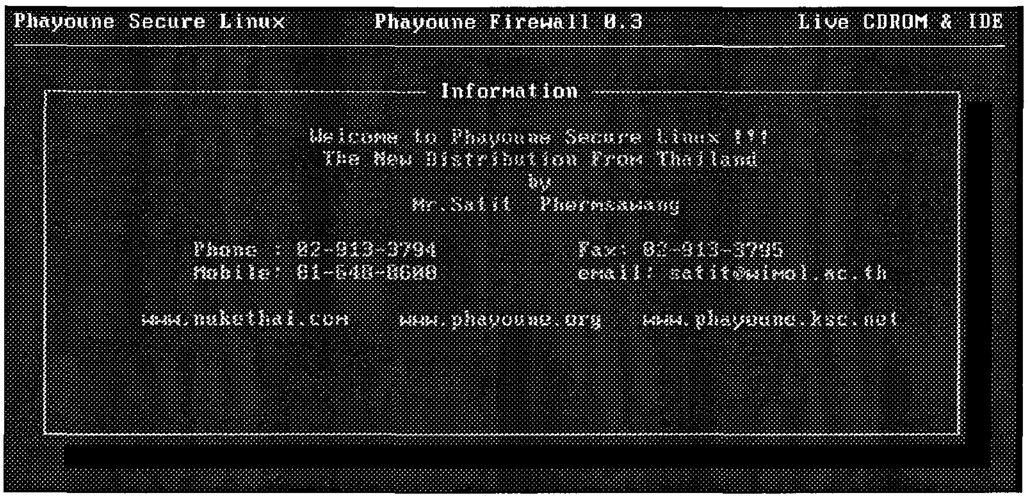
(2) Pre-Installation

(a) It is necessary to set the Bios to boot from the CD-Rom first, and the Floppy disk drive second, and hard drive third.

(b) Place the Phayoune CD in the CD-Rom drive

(c) Restart your computer

(d) Wait until the welcome screen appears on the computer display



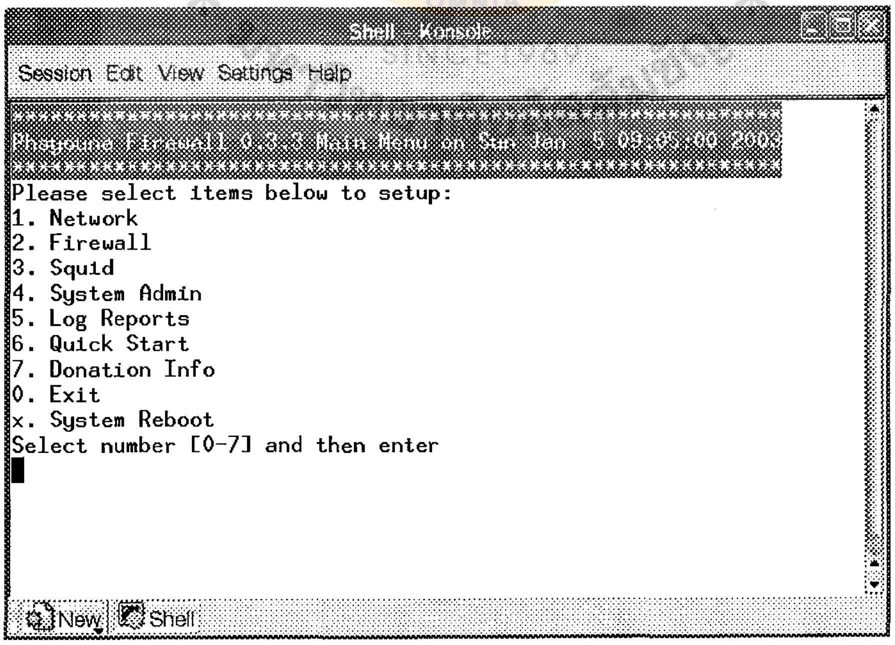
```
Phayoune Secure Linux        Phayoune Firewall 0.3            Live CDROM & IDE

                                 Information
                     Welcome to Phayoune Secure Linux !!!
                      The New Distribution From Thailand
                                     by
                           Mr.Satit  Phormsawang

          Phone : 02-913-3794             Fax: 02-913-3795
          Mobile: 01-648-0600             email: satit@wimol.ac.th

       www.nukethai.com    www.phayoune.org    www.phayoune.ksc.net
```

Figure 4.1. Phayoune Firewall Welcome Screen.

(e) Once you see the welcome screen, wait until you see the login: and password: prompts appear on bottom of the screen

(f) For the login, type in the word "root" not including the parenthesis

(g) For the password, type in the word "password" not including the parenthesis the main menu will appear

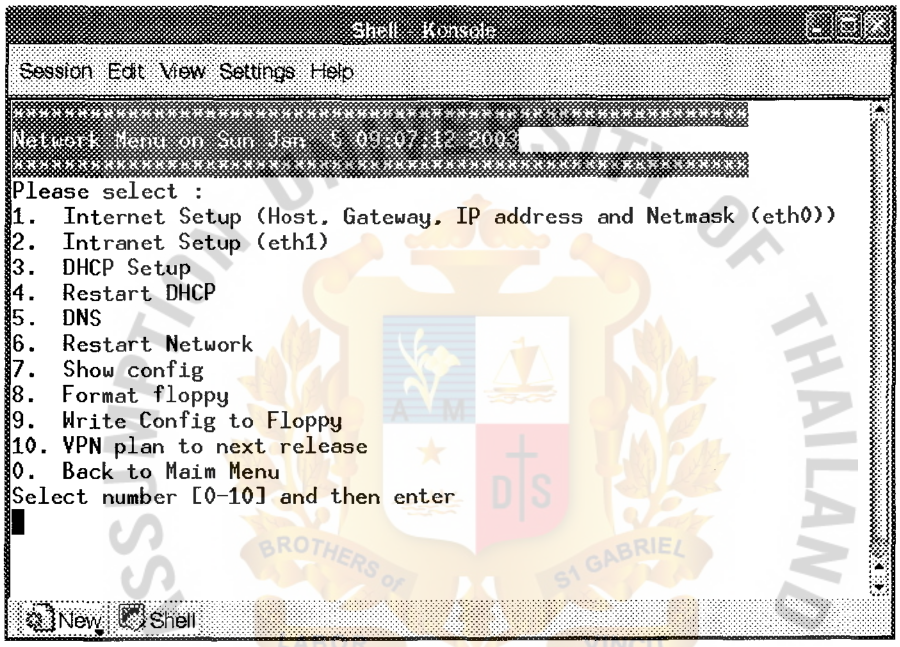


```
************************************************************
Phayoune Firewall 0.3.3 Main Menu on Sun Jan  5 09:05:00 2003
************************************************************
Please select items below to setup:
1. Network
2. Firewall
3. Squid
4. System Admin
5. Log Reports
6. Quick Start
7. Donation Info
0. Exit
x. System Reboot
Select number [0-7] and then enter
```

Figure 4.2. Main Menu.

4.2 Phayoune Firewall Menu Overview

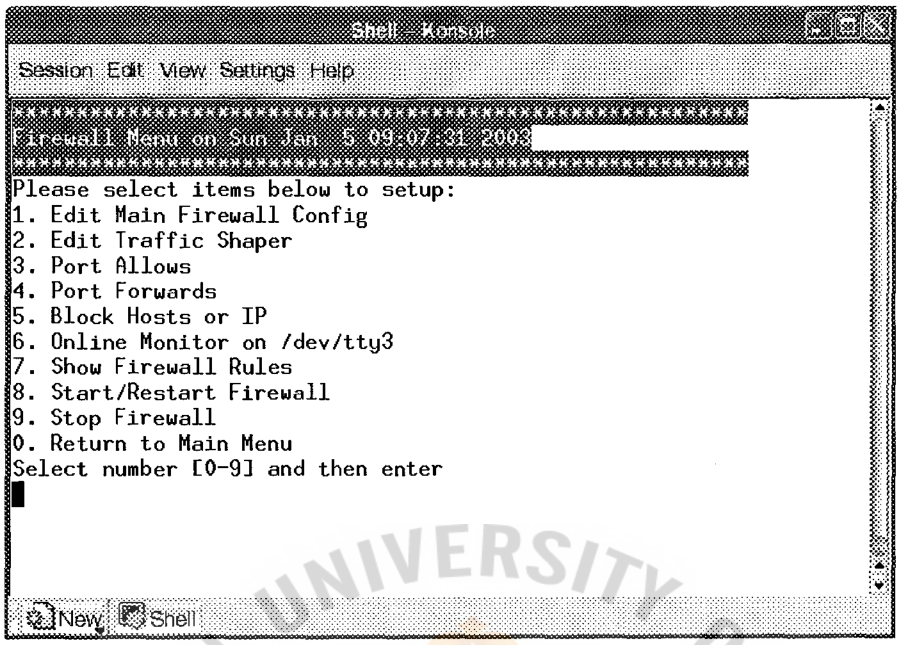
There are seven user menu's in this system. The first menu is the Network Menu. This menu will help you to set up many of the network features in the system. Such as your systems IP address, DHCP server, and DHCP clients. You may also use the menu to create a configuration backup diskette and other functions.



```
*************************************************************************
Network Menu on Sun Jan  5 09:07:12 2003
*************************************************************************
Please select :
1.  Internet Setup (Host, Gateway, IP address and Netmask (eth0))
2.  Intranet Setup (eth1)
3.  DHCP Setup
4.  Restart DHCP
5.  DNS
6.  Restart Network
7.  Show config
8.  Format floppy
9.  Write Config to Floppy
10. VPN plan to next release
0.  Back to Maim Menu
Select number [0-10] and then enter
```

Figure 4.3. Network Menu.

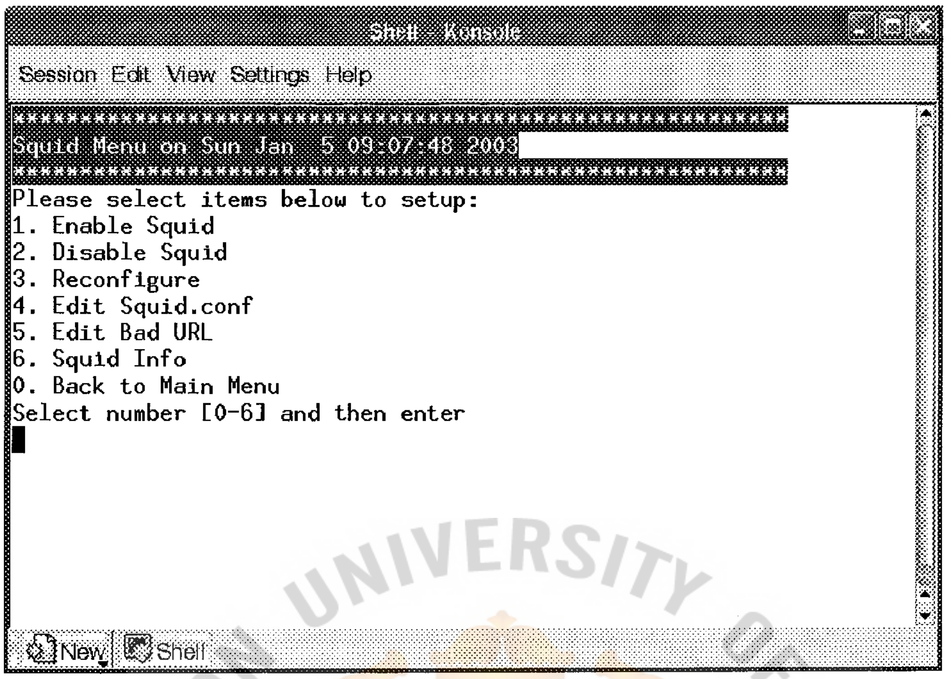
The second menu is the Firewall Menu. This menu helps you to closely set up the firewall policies. You can configure the web and Internet mail forwarding server. You can setup the system to allow all or specific groups in a network to have access behind the firewall.



```
*************************************************************
Firewall Menu on Sun Jan  5 09:07:31 2003
*************************************************************
Please select items below to setup:
1. Edit Main Firewall Config
2. Edit Traffic Shaper
3. Port Allows
4. Port Forwards
5. Block Hosts or IP
6. Online Monitor on /dev/tty3
7. Show Firewall Rules
8. Start/Restart Firewall
9. Stop Firewall
0. Return to Main Menu
Select number [0-9] and then enter
```

Figure 4.4. Firewall Menu.

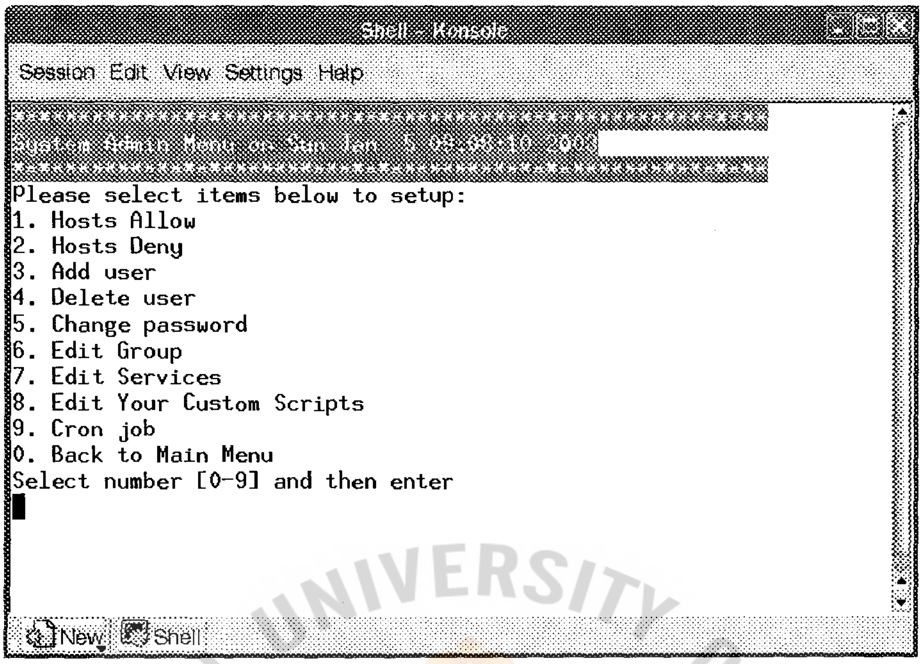
The third Menu is called the squid menu. This menu will help you to filter URL and worm viruses, such as Code Red, and Nimda viruses from infecting computers behind the firewall. From this menu you may also filter such URL's as pornographic sites, bomb building sites, or other types of sites unfit for general access.



```
**********************************************************
Squid Menu on Sun Jan  5 09:07:48 2003
**********************************************************
Please select items below to setup:
1. Enable Squid
2. Disable Squid
3. Reconfigure
4. Edit Squid.conf
5. Edit Bad URL
6. Squid Info
0. Back to Main Menu
Select number [0-6] and then enter
```

Figure 4.5. Squid Menu.

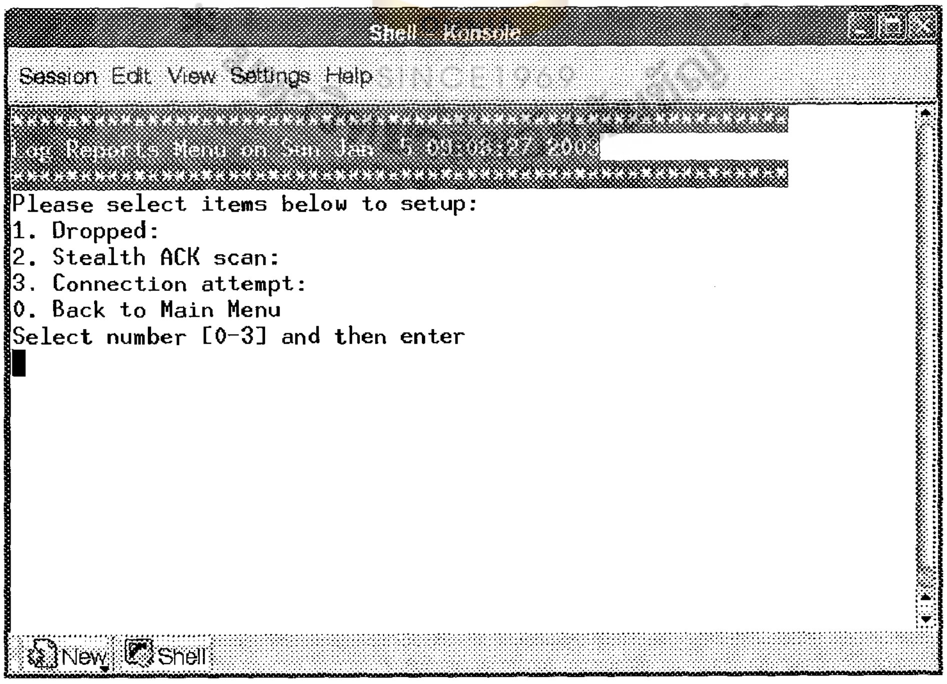
The fourth Menu is called the System Admin Menu. This menu will help you to manage all services in the system, such as mail service, web service, squid service, and DHCP server service. You can also add users and host for remote login from this menu, change user name and password, and change system admin user name and password. You can configure your cron job as well. The highlight of this menu is the user custom script. This feature makes it flexible for users to create their own scripts for specified jobs.



```
**********************************************************
System Admin Menu on Sun Jan  5 09:08:10 2003
**********************************************************
Please select items below to setup:
1. Hosts Allow
2. Hosts Deny
3. Add user
4. Delete user
5. Change password
6. Edit Group
7. Edit Services
8. Edit Your Custom Scripts
9. Cron job
0. Back to Main Menu
Select number [0-9] and then enter
```

Figure 4.6. System Admin Menu.

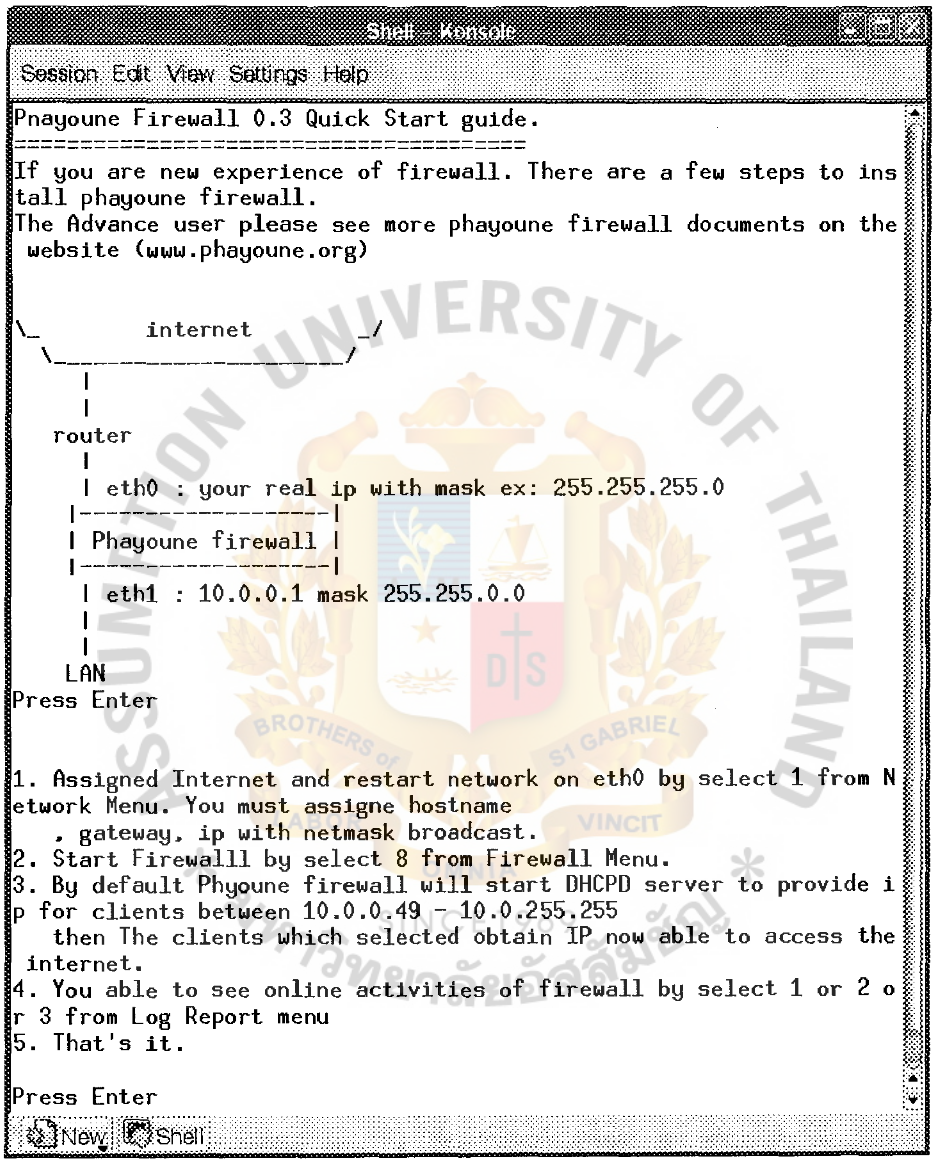
The fifth menu is the Log Report menu. This menu will show online attacks from systems outside of the firewall. You can select the intervals of report and select the email address to send the reports.



```
************************************************************
Log Reports Menu on Sun Jan  5 09:08:27 2003
************************************************************
Please select items below to setup:
1. Dropped:
2. Stealth ACK scan:
3. Connection attempt:
0. Back to Main Menu
Select number [0-3] and then enter
```

Figure 4.7. Log Reports Menu.

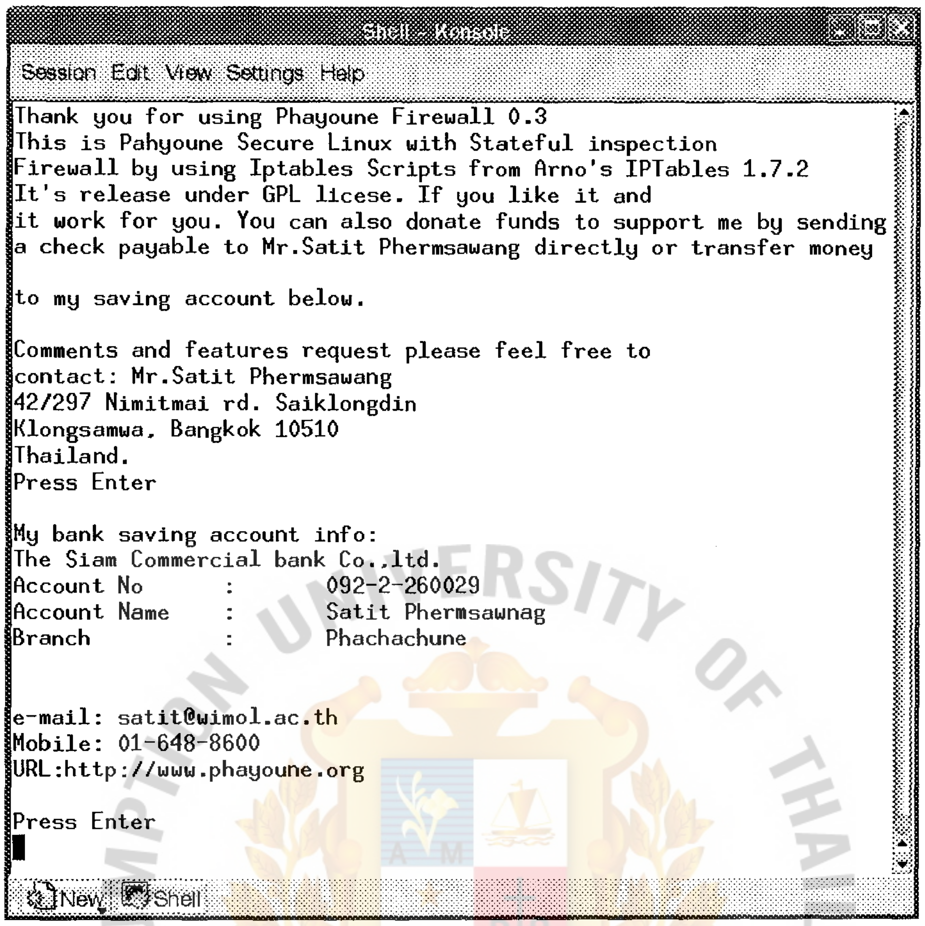
The sixth menu is the quick start menu. This menu is user friendly for users who are not that knowledgeable of a firewall system. The requirements needed are very basic to use the system.



```
Pnayoune Firewall 0.3 Quick Start guide.
======================================
If you are new experience of firewall. There are a few steps to ins
tall phayoune firewall.
The Advance user please see more phayoune firewall documents on the
 website (www.phayoune.org)


\_         internet          _/
  \_______________________/
      |
      |
   router
      |
      | eth0  : your real ip with mask ex: 255.255.255.0
     |-------------------|
     | Phayoune firewall |
     |-------------------|
      | eth1  : 10.0.0.1 mask 255.255.0.0
      |
      |
    LAN
Press Enter


1. Assigned Internet and restart network on eth0 by select 1 from N
etwork Menu. You must assigne hostname
   , gateway, ip with netmask broadcast.
2. Start Firewalll by select 8 from Firewall Menu.
3. By default Phyoune firewall will start DHCPD server to provide i
p for clients between 10.0.0.49 - 10.0.255.255
   then The clients which selected obtain IP now able to access the
 internet.
4. You able to see online activities of firewall by select 1 or 2 o
r 3 from Log Report menu
5. That's it.

Press Enter
```

Figure 4.8. Quick Start Menu.

The seventh menu is the donation menu. In this menu you will find the necessary information to help support the running and future development of this firewall system.



```
Thank you for using Phayoune Firewall 0.3
This is Pahyoune Secure Linux with Stateful inspection
Firewall by using Iptables Scripts from Arno's IPTables 1.7.2
It's release under GPL licese. If you like it and
it work for you. You can also donate funds to support me by sending
a check payable to Mr.Satit Phermsawang directly or transfer money

to my saving account below.

Comments and features request please feel free to
contact: Mr.Satit Phermsawang
42/297 Nimitmai rd. Saiklongdin
Klongsamwa, Bangkok 10510
Thailand.
Press Enter

My bank saving account info:
The Siam Commercial bank Co.,ltd.
Account No      :      092-2-260029
Account Name    :      Satit Phermsawnag
Branch          :      Phachachune


e-mail: satit@wimol.ac.th
Mobile: 01-648-8600
URL:http://www.phayoune.org

Press Enter
```

Figure 4.9. Donation Menu.

4.3 Quick Installation Guide

For quick installation, it is assumed that you need the functions of a firewall only. For this installation, there is no web server service, or mail server service.

There are a few configurations that you must setup in order to perform quick installation.

(a) Setup your network configurations such as IP address, Host Name, and Gateway, by selecting menu number 1 from the Main menu.

(b) Once you have selected menu number 1 from the main menu, you will see the Network menu. From the network menu, select menu number 1, Internet setup.

(c) Once you have selected the Internet setup menu, a blinking cursor will appear. At the blinking cursor, insert your host name. Example: FW1

(d) Press enter, a blinking cursor will appear, at this cursor, insert your gateway. Example: 203.155.105.254 The gateway is the IP address of your router.

(e) Press enter, a blinking cursor will appear, at this cursor, insert your IP address. Example: 203.155.105

(f) Press enter, a blinking cursor will appear, at this cursor, insert your Net Mask. Example: 255.255.255.0

(g) Press enter, a blinking cursor will appear, at this cursor, insert your Boardcast. Example: 203.155.105.255

(h) Press enter. Then press y for yes to start the network service.

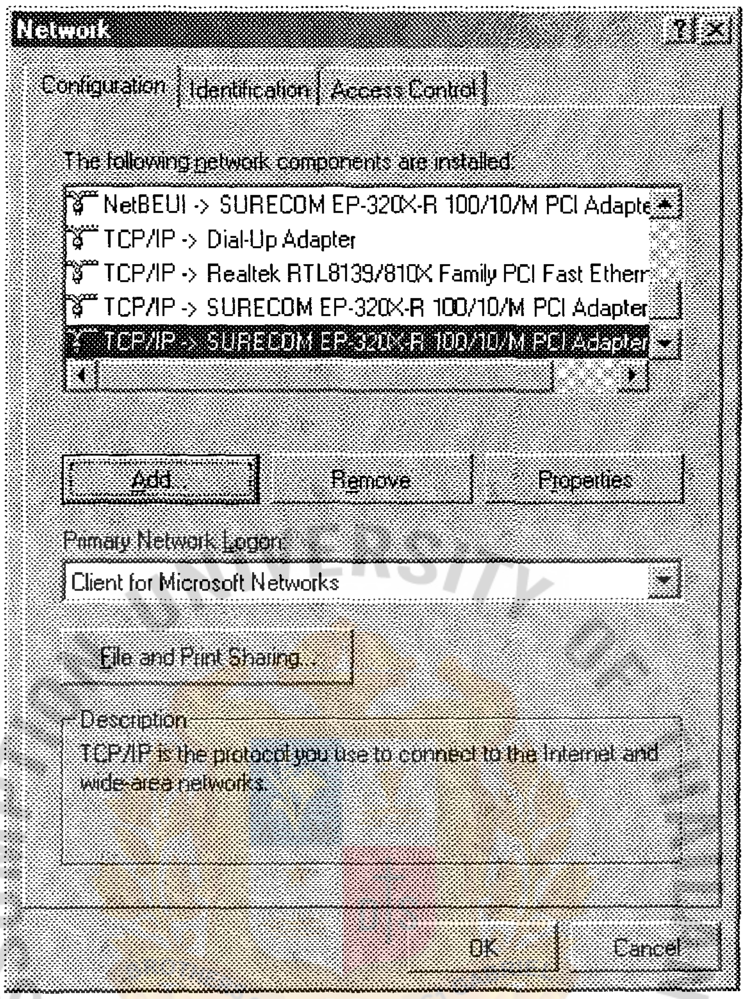
(i) Type 0 to go back to the main menu.

(j) Configure firewall information by selecting the firewall menu by pressing number 2.

(k) From the firewall menu, select menu number 8 to start the firewall service.

4.4 Clients Setup

For windows clients, click on the start button. Select setting. Select Control panel Select network. A network popup window will appear.



The following network components are installed:

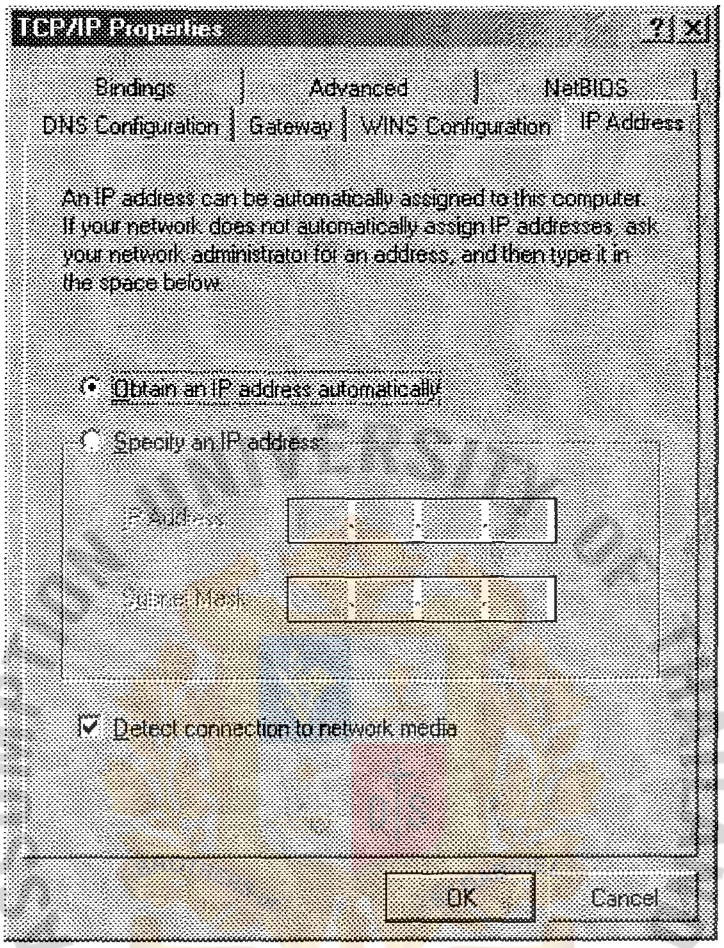
NetBEUI -> SURECOM EP-320X-R 100/10/M PCI Adapte
TCP/IP -> Dial-Up Adapter
TCP/IP -> Realtek RTL8139/810X Family PCI Fast Etherr
TCP/IP -> SURECOM EP-320X-R 100/10/M PCI Adapter
TCP/IP -> SURECOM EP-320X-R 100/10/M PCI Adapter

Primary Network Logon:
Client for Microsoft Networks

Description
TCP/IP is the protocol you use to connect to the Internet and wide-area networks.

Figure 4.10. Network Popup Menu.

Double click TCP/IP. The TCP/IP popup window will appear.


TCP/IP Properties

Bindings | Advanced | NetBIOS
DNS Configuration | Gateway | WINS Configuration | IP Address

An IP address can be automatically assigned to this computer. If your network does not automatically assign IP addresses, ask your network administrator for an address, and then type it in the space below.

(●) Obtain an IP address automatically
( ) Specify an IP address:
IP Address: . . .
Subnet Mask: . . .

[✓] Detect connection to network media

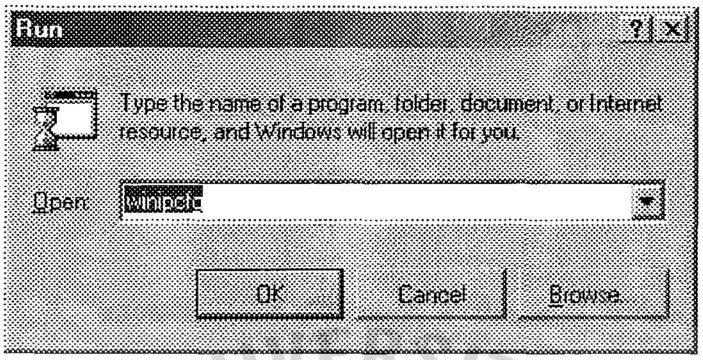
OK | Cancel

Figure 4.11. TCP/IP Properties.

Click Obtain an IP address automatically. Click OK at the bottom of the window. Close the TCP/IP popup window and return to the network popup window. Click OK on the bottom of this window.

The computer will ask you to reboot your computer. Choose reboot in order for the changes to take effect.

Once windows re-opens, you should test the system by selecting the start menu, and then selecting run. Type the command: WINIPCFG Press enter. The IP configuration popup window will appear. If your IP address resembles 10.x.x.x, it shows that you have properly installed the program. Now access the internet with your web browser. If all is functional, your setup is complete. If not, repeat each step again.



Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: winipcfg

Figure 4.12. RUN Popup Windows.

4.5 How to Make a Configuration Floppy

After your client can access the Internet smoothly, it is recommend to make a backup floppy. At this point, you have already configured your network and firewall. Most of your configurations are written on your RAM drive. If there is a system reboot, you will lose this information. It is important for you to make a backup configuration file to a floppy.

From the main menu, choose menu number 1 to open the network menu. Make sure you have inserted an empty diskette into the floppy drive. From the network menu, select menu number 8, to format your diskette. Then select menu number 9 to start the backup.

Write protect your diskette in order to protect the information that you have backed up. In order for the backup to take effect, the diskette must be placed in the floppy drive at all times.

V. CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The Building of a firewall requires a clear understanding of the networking requirements of a group. The installation is likely to have a direct impact on every machine behind the firewall. Since firewalls are tools used to implement network security policy, no firewall design should ever be considered without first clearly defining the ultimate security policy goals.

Risk assessment. The development of a security policy is driven by risk assessment and vulnerability analysis. A group needs to ask questions such as--"What are we trying to protect?" "Do we have anything of value on these computers?" "What would happen if we had a break- in?" If there is something of significant value on the network, then a risk assessment should be carried out. After a general risk assessment is performed, a more detailed inventory of open network services and potential vulnerabilities should follow.

It is not always the case that a firewall is the right tool for the job. If there are only a few computers within a group, then a firewall might be more than is required. Tightening host security is always a good idea, and in some cases is more than sufficient. Isolating sensitive hosts off the network can also be a practical solution in some cases. But in general, choosing a firewall may be appropriate when there is a valuable asset that is at risk, or there are so many computers being used within a group that being sure of good host security is not possible.

A general "posture" or "stance" is usually chosen for the security policy design. This stance is used as a starting point and a conceptual framework for guiding further development of the policy. Three of the most common postures are discussed below: trust inside, least privilege, and selective blocking.

Trust inside. The most popular stance is known as "trust inside." In this scenario, it is assumed that the most significant threats will come from outside the local area network, and the emphasis of the policy will be keeping outsiders from getting in. This type of stance is frequently implemented by defining a firewall rule set that permits all connections which are initiated from the inside, but blocks connections initiated from the outside. This type of policy is easy to conceptualize and fairly easy to implement and manage.

Least privilege. Another common stance is known as "least privilege." In this stance, it is assumed that all network connections are blocked in both directions as a starting point, and the policy is incrementally opened to define precisely what is allowed. This is also known as the "deny everything" stance. Some of the individual-PC software firewalls operate in this manner, for example "ZoneAlarm Pro" and "Sygate Personal Firewall." These products force you to define firewall rules for each and every network application that is used for an outgoing connection, and also force you to define firewall rules upon receipt of new incoming connections.

Selective blocking. "Selective blocking" is another common posture. This is also known as an "accept everything" starting point. The policy is fine tuned by explicitly denying only selected connections which are known to be potentially dangerous. This is clearly the most vulnerable stance to use as a starting point. Selective blocking is often used as a first line of defense. One example is the blocking of selected incoming ports using packet filtering on a border router.

The type of firewall you choose will depend not only on the policy goals, but also on the resources available to your organization and the performance requirements of the device. Other obvious considerations are the availability of VPN software, routing and addressing features, the ease of use of the management interface, costs for support maintenance, availability of hardware redundancy, and so on.

Firewalls are typically categorized into a few types: packet filters, stateful inspection firewalls (ex. Phayoune Firewall), and application proxies. Stateful inspection firewalls, which are sometimes referred to as "session-based firewalls," have become more popular in recent years. The development of hardware redundancy including session fail over is a more recent feature. Every firewall product may have some or all of these general firewall capabilities.

Hardware appliances, such as the Netscreen-204, Cisco PIX 515, and Sonicwall Pro, continue to be popular because of their ease-of-use and low maintenance requirements. Since these devices do not have hard disk drives, they tend to have a high degree of hardware reliability. More expensive products, such as the Netscreen 500 and Cisco PIX 535, support higher speed interfaces at higher cost.

Do-it-yourself Unix-based firewalls by running iptables, ipchains, or ipfilter, can also perform well. This project recommended to install Phayoune Firewall which I had made with securities in mind. Which is release under GPL. Phayoune Firewall is a Thai Linux distribution that is intended to be used as a stateful inspection firewall. It runs directly from CD- ROM and does not require a hard drive. It supports port forwarding, an online monitor, Squid blocking for worms, virii, and adult content, and a menu-based setup that stores configuration details on a floppy.

5.2 Recommendations

A firewall is an important security feature for any Internet user. Available in software or standalone hardware forms, a firewall prevents unauthorized users and/or data from getting in or out of your network, using rules to specify acceptable communications from locations, individuals, or in certain protocols. However, firewalls do not protect your data from threats within the Internet network itself. Once the data gets outside your firewall, your user names, passwords, account numbers, server addresses, and other sensitive information are visible to hackers. VPN (Virture Private Network) as tunnels, enabled by encryption algorithms, give you the ability to use the public, shared Internet for secure data transmission after it leaves the protective custody of your firewall.

Although, Phayoune Firewall design for dual-homed firewall solution is capable of routing packet between the two-network interfaces. I plan to add more features such as Graphic menu Interface, VPN, Virtual Private Network with IPSEC from freeswan, in the next release. Be patient and keep trace on www.phayoune.org.

In addition to firewalls, you also look at intrusion detection software meant for the at-home user, such as BlackICE and ZoneAlarm, that help detect network attacks as they happen. The sooner you known an attack is taking place, the sooner you can react to the attack and minimize the damage that attacker inflicts.

APPENDIX A

GLOSSARY

GLOSSARY

Firewall - a device used to implement a security policy between networks. A firewall has multiple network interfaces, and is typically used to create a secure boundary between untrusted external networks and trusted internal networks. The security policy defines what type of access is allowed between the connected networks.

DMZ - demilitarized zone, a perimeter network established to house public services, maintained outside of the internal/protected network. Since a DMZ will be open to allow public access to services, it is considered less secure than the internal/protected network.

Private Addresses - a specified set of IP address ranges that have been set aside for use within internal networks. These addresses begin with the following prefixes: 10.x.x.x, 192.168.x.x, and 172.16.x.x-172.31.x.x. Organizations use these addresses internally but do not forward them beyond their routers. See RFC 1597 at http://www.nic.mil/ftp/rfc/rfc1597.txt

NAT - network address translation, a method by which IP addresses are mapped transparently from one group of addresses to another. If ports are also mapped, this may be referred to as "port address translation." NAT is often used to map a set of private/internal addresses to a smaller set of public/external addresses. See RFC 1631 at http://www.nic.mil/ftp/rfc/rfc1631.txt

VPN - virtual private network, a secure network connection built on top of an existing public network. VPNs are used to connect remote clients or remote branch offices to local protected networks using secure encryption and authentication technologies.

DHCP - Dynamic Host Configuration Protocol. This is an upgrade of an old standard, BOOTP, that has survived the introduction of windows and other tasks not originally envisioned for it. The principal is that a device that knows nothing about its own network settings sends out a broadcast packet saying, in effect, tell me what to do. The DHCP server is listening for these packets, and responds with a packet containing the settings that device should have. A DHCP server is configured with a table of ethernet addresses, and ranges of IP addresses, and maps that describe who gets what and when. Because a constantly changing IP address can be destabilizing, the DHCP server uses the concept of a lease, to tell the device it has this IP address for only a limited time. When the lease runs out, the device tries to obtain a renewal and, possibly, gets a new IP address given to it! If the device is allocated a new IP address, it must reconfigure itself to take that feedback needed:please tell us whether windows machines can handle this or reboot themselves.

APPENDIX B

PHAYOUNE FIREWALL DEFAULT POLICIES

PHAYOUNE FIREWALL DEFAULT POLICIES

Chain INPUT (policy DROP)

```
target     prot opt source              destination

ACCEPT     all  --  0.0.0.0/0           0.0.0.0/0

LOG        all  --  192.168.0.0/24      0.0.0.0/0          limit: avg 3/min burst 5 LOG flags 0

level 6 prefix `Spoofed (local) packet: '

DROP       all  --  192.168.0.0/24      0.0.0.0/0

LOG        icmp --  0.0.0.0/0           0.0.0.0/0          state INVALID limit: avg 3/min burst

2 LOG flags 0 level 6 prefix `INVALID INPUT packet: '

LOG        !icmp --  0.0.0.0/0          0.0.0.0/0          state INVALID limit: avg 3/min burst

2 LOG flags 0 level 6 prefix `INVALID INPUT packet: '

DROP       all  --  0.0.0.0/0           0.0.0.0/0          state INVALID

HOST_BLOCK  all  --  0.0.0.0/0          0.0.0.0/0

ACCEPT     all  --  0.0.0.0/0           0.0.0.0/0          state ESTABLISHED

ACCEPT     all  --  0.0.0.0/0           0.0.0.0/0

VALID_CHECK  all  --  0.0.0.0/0         0.0.0.0/0

ACCEPT     all  --  0.0.0.0/0           0.0.0.0/0          state RELATED

CHECK      !icmp --  0.0.0.0/0          0.0.0.0/0          state NEW

CHECK      icmp --  0.0.0.0/0           0.0.0.0/0          state NEW limit: avg 10/sec burst

50

LOG        icmp --  0.0.0.0/0           0.0.0.0/0          icmp type 8 limit: avg 12/hour burst 1

LOG flags 0 level 6 prefix `ICMP flood: '

LOG        all  --  0.0.0.0/0           0.0.0.0/0          limit: avg 1/sec burst 5 LOG flags 0

level 6 prefix `Dropped INPUT packet: '

DROP       all  --  0.0.0.0/0           0.0.0.0/0
```

```
Chain FORWARD (policy DROP)
target     prot opt source               destination
TCPMSS     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp flags:0x06/0x02 TCPMSS clamp
to PMTU
LOG        icmp --  0.0.0.0/0            0.0.0.0/0           state INVALID limit: avg 3/min burst
2 LOG flags 0 level 6 prefix `INVALID FORWARD packet: '
LOG        !icmp --  0.0.0.0/0           0.0.0.0/0           state INVALID limit: avg 3/min burst
2 LOG flags 0 level 6 prefix `INVALID FORWARD packet: '
DROP       all  --  0.0.0.0/0            0.0.0.0/0           state INVALID
HOST_BLOCK  all  --  0.0.0.0/0           0.0.0.0/0
DROP       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpts:6666:6669
DROP       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp spts:6666:6669
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state ESTABLISHED
VALID_CHECK  all  --  0.0.0.0/0          0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED
ACCEPT     all  --  192.168.0.0/24       0.0.0.0/0           state NEW
DROP       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:80 flags:!0x16/0x02 state
NEW
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:80 flags:0x16/0x02 state
NEW
DROP       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:443 flags:!0x16/0x02 state
NEW
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:443 flags:0x16/0x02 state
NEW
```

```
ACCEPT    udp  --  0.0.0.0/0          0.0.0.0/0        udp dpt:80 state NEW

ACCEPT    udp  --  0.0.0.0/0          0.0.0.0/0        udp dpt:443 state NEW

LOG       all  --  0.0.0.0/0          0.0.0.0/0        limit: avg 1/sec burst 5 LOG flags 0

level 6 prefix `Dropped FORWARD packet: '

DROP      all  --  0.0.0.0/0          0.0.0.0/0
```

Chain OUTPUT (policy ACCEPT)

```
target     prot opt source             destination

TCPMSS     tcp  --  0.0.0.0/0          0.0.0.0/0        tcp flags:0x06/0x02 TCPMSS clamp

to PMTU

HOST_BLOCK  all  --  0.0.0.0/0         0.0.0.0/0

LOG        all  -f  0.0.0.0/0          0.0.0.0/0        limit: avg 3/min burst 5 LOG flags 0

level 6 prefix `FRAGMENTED PACKET (OUT): '

DROP       all  -f  0.0.0.0/0          0.0.0.0/0
```

Chain CHECK (2 references)

```
target     prot opt source             destination

LOG        tcp  --  0.0.0.0/0          0.0.0.0/0        tcp dpt:0 limit: avg 1/hour burst 1 LOG

flags 0 level 6 prefix `TCP port 0 OS fingerprint: '

LOG        udp  --  0.0.0.0/0          0.0.0.0/0        udp dpt:0 limit: avg 1/hour burst 1

LOG flags 0 level 6 prefix `UDP port 0 OS fingerprint: '

DROP       tcp  --  0.0.0.0/0          0.0.0.0/0        tcp dpt:0

DROP       udp  --  0.0.0.0/0          0.0.0.0/0        udp dpt:0

DROP       tcp  --  203.155.101.0/24   0.0.0.0/0        tcp dpt:22 flags:!0x16/0x02
```

```
ACCEPT    tcp  --  203.155.101.0/24    0.0.0.0/0      tcp dpt:22 flags:0x16/0x02
DROP      tcp  --  203.155.101.0/24    0.0.0.0/0      tcp dpt:8080 flags:!0x16/0x02
ACCEPT    tcp  --  203.155.101.0/24    0.0.0.0/0      tcp dpt:8080 flags:0x16/0x02
DROP      tcp  --  203.155.100.0/24    0.0.0.0/0      tcp dpt:22 flags:!0x16/0x02
ACCEPT    tcp  --  203.155.100.0/24    0.0.0.0/0      tcp dpt:22 flags:0x16/0x02
DROP      tcp  --  203.155.100.0/24    0.0.0.0/0      tcp dpt:8080 flags:!0x16/0x02
ACCEPT    tcp  --  203.155.100.0/24    0.0.0.0/0      tcp dpt:8080 flags:0x16/0x02
DROP      tcp  --  203.155.105.0/24    0.0.0.0/0      tcp dpt:22 flags:!0x16/0x02
ACCEPT    tcp  --  203.155.105.0/24    0.0.0.0/0      tcp dpt:22 flags:0x16/0x02
ACCEPT    udp  --  203.155.101.0/24    0.0.0.0/0      udp dpt:22
ACCEPT    udp  --  203.155.101.0/24    0.0.0.0/0      udp dpt:8080
ACCEPT    udp  --  203.155.100.0/24    0.0.0.0/0      udp dpt:22
ACCEPT    udp  --  203.155.100.0/24    0.0.0.0/0      udp dpt:8080
ACCEPT    udp  --  203.155.105.0/24    0.0.0.0/0      udp dpt:22
LOG       icmp --  0.0.0.0/0           0.0.0.0/0      limit: avg 3/min burst 1 LOG flags 0
level 6 prefix `Dropped ICMP packet: '
DROP      tcp  --  0.0.0.0/0           0.0.0.0/0      tcp spts:20:9999 dpts:1024:65535
flags:!0x16/0x02 limit: avg 10/sec burst 25
DROP      udp  --  0.0.0.0/0           0.0.0.0/0      udp spts:20:9999 dpts:1024:65535
limit: avg 10/sec burst 25
LOG       tcp  --  0.0.0.0/0           0.0.0.0/0      tcp spts:20:9999 dpts:1024:65535
flags:!0x16/0x02 limit: avg 6/hour burst 1 LOG flags 0 level 6 prefix `Lost TCP
connection flood?: '
LOG       udp  --  0.0.0.0/0           0.0.0.0/0      udp spts:20:9999 dpts:1024:65535
limit: avg 6/hour burst 1 LOG flags 0 level 6 prefix `Lost UDP connection flood?: '
```

```
DROP      tcp  --  0.0.0.0/0          0.0.0.0/0          tcp spts:20:9999 dpts:1024:65535
flags:!0x16/0x02
DROP      udp  --  0.0.0.0/0          0.0.0.0/0          udp spts:20:9999 dpts:1024:65535
LOG       tcp  --  0.0.0.0/0          0.0.0.0/0          tcp dpts:1024:65535 flags:!0x16/0x02
limit: avg 3/min burst 5 LOG flags 0 level 6 prefix `Stealth scan (UNPRIV)?: '
LOG       tcp  --  0.0.0.0/0          0.0.0.0/0          tcp dpts:0:1023 flags:!0x16/0x02 limit:
avg 3/min burst 5 LOG flags 0 level 6 prefix `Stealth scan (PRIV)?: '
DROP      tcp  --  0.0.0.0/0          0.0.0.0/0          tcp flags:!0x16/0x02
LOG       tcp  --  0.0.0.0/0          0.0.0.0/0          tcp dpts:0:1023 limit: avg 2/min burst 2
LOG flags 0 level 6 prefix `Connection attempt (PRIV): '
LOG       udp  --  0.0.0.0/0          0.0.0.0/0          udp dpts:0:1023 limit: avg 2/min burst
2 LOG flags 0 level 6 prefix `Connection attempt (PRIV): '
LOG       tcp  --  0.0.0.0/0          0.0.0.0/0          tcp dpts:1024:65535 limit: avg 1/min
burst 1 LOG flags 0 level 6 prefix `Connection attempt (UNPRIV): '
LOG       udp  --  0.0.0.0/0          0.0.0.0/0          udp dpts:1024:65535 limit: avg 1/min
burst 1 LOG flags 0 level 6 prefix `Connection attempt (UNPRIV): '
DROP      all  --  0.0.0.0/0          0.0.0.0/0
```

Chain HOST_BLOCK (3 references)

```
target     prot opt source               destination
Chain VALID_CHECK (2 references)
target     prot opt source               destination
LOG       tcp  --  0.0.0.0/0          0.0.0.0/0          tcp flags:0x3F/0x29 limit: avg 3/min
burst 5 LOG flags 0 level 6 prefix `Stealth XMAS scan: '
```

```
LOG      tcp  --  0.0.0.0/0          0.0.0.0/0         tcp flags:0x3F/0x37 limit: avg 3/min
burst 5 LOG flags 0 level 6 prefix `Stealth XMAS-PSH scan: '
LOG      tcp  --  0.0.0.0/0          0.0.0.0/0         tcp flags:0x3F/0x3F limit: avg 3/min
burst 5 LOG flags 0 level 6 prefix `Stealth XMAS-ALL scan: '
LOG      tcp  --  0.0.0.0/0          0.0.0.0/0         tcp flags:0x3F/0x01 limit: avg 3/min
burst 5 LOG flags 0 level 6 prefix `Stealth FIN scan: '
LOG      tcp  --  0.0.0.0/0          0.0.0.0/0         tcp flags:0x06/0x06 limit: avg 3/min
burst 5 LOG flags 0 level 6 prefix `Stealth SYN/RST scan: '
LOG      tcp  --  0.0.0.0/0          0.0.0.0/0         tcp flags:0x03/0x03 limit: avg 3/min
burst 5 LOG flags 0 level 6 prefix `Stealth SYN/FIN scan(?): '
LOG      tcp  --  0.0.0.0/0          0.0.0.0/0         tcp flags:0x3F/0x00 limit: avg 3/min
burst 5 LOG flags 0 level 6 prefix `Stealth Null scan: '
DROP     tcp  --  0.0.0.0/0         0.0.0.0/0        tcp flags:0x3F/0x29
DROP     tcp  --  0.0.0.0/0         0.0.0.0/0        tcp flags:0x3F/0x37
DROP     tcp  --  0.0.0.0/0         0.0.0.0/0        tcp flags:0x3F/0x3F
DROP     tcp  --  0.0.0.0/0         0.0.0.0/0        tcp flags:0x3F/0x01
DROP     tcp  --  0.0.0.0/0         0.0.0.0/0        tcp flags:0x06/0x06
DROP     tcp  --  0.0.0.0/0         0.0.0.0/0        tcp flags:0x03/0x03
DROP     tcp  --  0.0.0.0/0         0.0.0.0/0        tcp flags:0x3F/0x00
LOG      tcp  --  0.0.0.0/0          0.0.0.0/0        tcp option=64 limit: avg 3/min burst 1
LOG flags 0 level 6 prefix `Bad TCP flag(64): '
LOG      tcp  --  0.0.0.0/0          0.0.0.0/0        tcp option=128 limit: avg 3/min burst 1
LOG flags 0 level 6 prefix `Bad TCP flag(128): '
DROP     tcp  --  0.0.0.0/0         0.0.0.0/0        tcp option=64
DROP     tcp  --  0.0.0.0/0         0.0.0.0/0        tcp option=128
```

```
LOG       all  -f  0.0.0.0/0           0.0.0.0/0          limit: avg 3/min burst 1 LOG flags 0
level 4 prefix `Fragmented packet: '
DROP      all  -f  0.0.0.0/0           0.0.0.0/0
LOG       all  --  10.0.0.0/8          0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Class A address: '
LOG       all  --  172.16.0.0/12       0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Class B address: '
LOG       all  --  192.168.0.0/16      0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Class C address: '
LOG       all  --  169.254.0.0/16      0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Class M$ address: '
LOG       all  --  0.0.0.0/8           0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG       all  --  1.0.0.0/8           0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG       all  --  2.0.0.0/8           0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG       all  --  5.0.0.0/8           0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG       all  --  7.0.0.0/8           0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG       all  --  23.0.0.0/8          0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG       all  --  27.0.0.0/8          0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
```

```
LOG      all  --  31.0.0.0/8       0.0.0.0/0       limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG      all  --  36.0.0.0/8       0.0.0.0/0       limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG      all  --  37.0.0.0/8       0.0.0.0/0       limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG      all  --  39.0.0.0/8       0.0.0.0/0       limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG      all  --  41.0.0.0/8       0.0.0.0/0       limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG      all  --  42.0.0.0/8       0.0.0.0/0       limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG      all  --  58.0.0.0/8       0.0.0.0/0       limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG      all  --  59.0.0.0/8       0.0.0.0/0       limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG      all  --  60.0.0.0/8       0.0.0.0/0       limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG      all  --  69.0.0.0/8       0.0.0.0/0       limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG      all  --  70.0.0.0/8       0.0.0.0/0       limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG      all  --  71.0.0.0/8       0.0.0.0/0       limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
```

```
LOG        all  --  72.0.0.0/8         0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  73.0.0.0/8         0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  74.0.0.0/8         0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  75.0.0.0/8         0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  76.0.0.0/8         0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  77.0.0.0/8         0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  78.0.0.0/8         0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  79.0.0.0/8         0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  82.0.0.0/8         0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  83.0.0.0/8         0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  84.0.0.0/8         0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  85.0.0.0/8         0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
```

```
LOG        all  --  86.0.0.0/8          0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  87.0.0.0/8          0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  88.0.0.0/8          0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  89.0.0.0/8          0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  90.0.0.0/8          0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  91.0.0.0/8          0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  92.0.0.0/8          0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  93.0.0.0/8          0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  94.0.0.0/8          0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  95.0.0.0/8          0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  96.0.0.0/8          0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  97.0.0.0/8          0.0.0.0/0          limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
```

```
LOG        all  --  98.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  99.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  100.0.0.0/8       0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  101.0.0.0/8       0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  102.0.0.0/8       0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  103.0.0.0/8       0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  104.0.0.0/8       0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  105.0.0.0/8       0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  106.0.0.0/8       0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  107.0.0.0/8       0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  108.0.0.0/8       0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  109.0.0.0/8       0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
```

```
LOG        all  --  110.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  111.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  112.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  113.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  114.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  115.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  116.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  117.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  118.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  119.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  120.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  121.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
```

```
LOG        all  --  122.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  123.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  124.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  125.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  126.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  127.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  197.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  219.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  220.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  221.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  222.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  223.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
```

```
LOG    all -- 224.0.0.0/8    0.0.0.0/0    limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG    all -- 225.0.0.0/8    0.0.0.0/0    limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG    all -- 226.0.0.0/8    0.0.0.0/0    limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG    all -- 227.0.0.0/8    0.0.0.0/0    limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG    all -- 228.0.0.0/8    0.0.0.0/0    limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG    all -- 229.0.0.0/8    0.0.0.0/0    limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG    all -- 230.0.0.0/8    0.0.0.0/0    limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG    all -- 231.0.0.0/8    0.0.0.0/0    limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG    all -- 232.0.0.0/8    0.0.0.0/0    limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG    all -- 233.0.0.0/8    0.0.0.0/0    limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG    all -- 234.0.0.0/8    0.0.0.0/0    limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG    all -- 235.0.0.0/8    0.0.0.0/0    limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
```

```
LOG        all  --  236.0.0.0/8        0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  237.0.0.0/8        0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  238.0.0.0/8        0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  239.0.0.0/8        0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  240.0.0.0/8        0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  241.0.0.0/8        0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  242.0.0.0/8        0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  243.0.0.0/8        0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  244.0.0.0/8        0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  245.0.0.0/8        0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  246.0.0.0/8        0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  247.0.0.0/8        0.0.0.0/0         limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
```

```
LOG        all  --  248.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  249.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  250.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  251.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  252.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  253.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  254.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
LOG        all  --  255.0.0.0/8        0.0.0.0/0        limit: avg 1/min burst 1 LOG flags 0
level 6 prefix `Reserved address: '
DROP       all  --  10.0.0.0/8       0.0.0.0/0
DROP       all  --  172.16.0.0/12    0.0.0.0/0
DROP       all  --  192.168.0.0/16  0.0.0.0/0
DROP       all  --  169.254.0.0/16  0.0.0.0/0
DROP       all  --  0.0.0.0/8        0.0.0.0/0
DROP       all  --  1.0.0.0/8        0.0.0.0/0
DROP       all  --  2.0.0.0/8        0.0.0.0/0
DROP       all  --  5.0.0.0/8        0.0.0.0/0
DROP       all  --  7.0.0.0/8        0.0.0.0/0
DROP      all  --  23.0.0.0/8        0.0.0.0/0
DROP      all  --  27.0.0.0/8        0.0.0.0/0
DROP      all  --  31.0.0.0/8        0.0.0.0/0
DROP      all  --  36.0.0.0/8        0.0.0.0/0
DROP      all  --  37.0.0.0/8        0.0.0.0/0
DROP      all  --  39.0.0.0/8        0.0.0.0/0
DROP      all  --  41.0.0.0/8        0.0.0.0/0
DROP      all  --  42.0.0.0/8        0.0.0.0/0
DROP      all  --  58.0.0.0/8        0.0.0.0/0
DROP      all  --  59.0.0.0/8        0.0.0.0/0
DROP      all  --  60.0.0.0/8        0.0.0.0/0
DROP      all  --  69.0.0.0/8        0.0.0.0/0
DROP      all  --  70.0.0.0/8        0.0.0.0/0
DROP      all  --  71.0.0.0/8        0.0.0.0/0
DROP      all  --  72.0.0.0/8        0.0.0.0/0
DROP      all  --  73.0.0.0/8         0.0.0.0/0
DROP      all  --  74.0.0.0/8         0.0.0.0/0
DROP      all  --  75.0.0.0/8         0.0.0.0/0
DROP      all  --  76.0.0.0/8         0.0.0.0/0
DROP      all  --  77.0.0.0/8         0.0.0.0/0
DROP      all  --  78.0.0.0/8         0.0.0.0/0
DROP      all  --  79.0.0.0/8         0.0.0.0/0
DROP      all  --  82.0.0.0/8         0.0.0.0/0
DROP      all  --  83.0.0.0/8         0.0.0.0/0
DROP      all  --  84.0.0.0/8         0.0.0.0/0
DROP      all  --  85.0.0.0/8       0.0.0.0/0
DROP      all  --  86.0.0.0/8       0.0.0.0/0
DROP      all  --  87.0.0.0/8       0.0.0.0/0
DROP      all  --  88.0.0.0/8       0.0.0.0/0
DROP      all  --  89.0.0.0/8       0.0.0.0/0
DROP      all  --  90.0.0.0/8       0.0.0.0/0
DROP      all  --  91.0.0.0/8       0.0.0.0/0
DROP      all  --  92.0.0.0/8       0.0.0.0/0
DROP      all  --  93.0.0.0/8       0.0.0.0/0
DROP      all  --  94.0.0.0/8       0.0.0.0/0
DROP      all  --  95.0.0.0/8       0.0.0.0/0
DROP      all  --  96.0.0.0/8       0.0.0.0/0
DROP      all  --  97.0.0.0/8       0.0.0.0/0
DROP      all  --  98.0.0.0/8       0.0.0.0/0
DROP      all  --  99.0.0.0/8       0.0.0.0/0
DROP      all  --  100.0.0.0/8      0.0.0.0/0
DROP      all  --  101.0.0.0/8      0.0.0.0/0
DROP      all  --  102.0.0.0/8      0.0.0.0/0
DROP      all  --  103.0.0.0/8      0.0.0.0/0
DROP      all  --  104.0.0.0/8      0.0.0.0/0
DROP      all  --  105.0.0.0/8      0.0.0.0/0
DROP      all  --  106.0.0.0/8      0.0.0.0/0
DROP      all  --  107.0.0.0/8      0.0.0.0/0
DROP      all  --  108.0.0.0/8      0.0.0.0/0
DROP      all  --  109.0.0.0/8       0.0.0.0/0
DROP    all -- 110.0.0.0/8      0.0.0.0/0
DROP    all -- 111.0.0.0/8      0.0.0.0/0
DROP    all -- 112.0.0.0/8      0.0.0.0/0
DROP    all -- 113.0.0.0/8      0.0.0.0/0
DROP    all -- 114.0.0.0/8      0.0.0.0/0
DROP    all -- 115.0.0.0/8      0.0.0.0/0
DROP    all -- 116.0.0.0/8      0.0.0.0/0
DROP    all -- 117.0.0.0/8      0.0.0.0/0
DROP    all -- 118.0.0.0/8      0.0.0.0/0
DROP    all -- 119.0.0.0/8      0.0.0.0/0
DROP    all -- 120.0.0.0/8      0.0.0.0/0
DROP    all -- 121.0.0.0/8      0.0.0.0/0
DROP    all -- 122.0.0.0/8      0.0.0.0/0
DROP    all -- 123.0.0.0/8      0.0.0.0/0
DROP    all -- 124.0.0.0/8      0.0.0.0/0
DROP    all -- 125.0.0.0/8      0.0.0.0/0
DROP    all -- 126.0.0.0/8      0.0.0.0/0
DROP    all -- 127.0.0.0/8      0.0.0.0/0
DROP    all -- 197.0.0.0/8      0.0.0.0/0
DROP    all -- 219.0.0.0/8      0.0.0.0/0
DROP    all -- 220.0.0.0/8      0.0.0.0/0
DROP    all -- 221.0.0.0/8      0.0.0.0/0
DROP    all -- 222.0.0.0/8      0.0.0.0/0
DROP    all -- 223.0.0.0/8      0.0.0.0/0
DROP    all -- 224.0.0.0/8      0.0.0.0/0
DROP      all  --  225.0.0.0/8      0.0.0.0/0
DROP      all  --  226.0.0.0/8      0.0.0.0/0
DROP      all  --  227.0.0.0/8      0.0.0.0/0
DROP      all  --  228.0.0.0/8      0.0.0.0/0
DROP      all  --  229.0.0.0/8      0.0.0.0/0
DROP      all  --  230.0.0.0/8      0.0.0.0/0
DROP      all  --  231.0.0.0/8      0.0.0.0/0
DROP      all  --  232.0.0.0/8      0.0.0.0/0
DROP      all  --  233.0.0.0/8      0.0.0.0/0
DROP      all  --  234.0.0.0/8      0.0.0.0/0
DROP      all  --  235.0.0.0/8      0.0.0.0/0
DROP      all  --  236.0.0.0/8      0.0.0.0/0
DROP      all  --  237.0.0.0/8      0.0.0.0/0
DROP      all  --  238.0.0.0/8      0.0.0.0/0
DROP      all  --  239.0.0.0/8      0.0.0.0/0
DROP      all  --  240.0.0.0/8      0.0.0.0/0
DROP      all  --  241.0.0.0/8      0.0.0.0/0
DROP      all  --  242.0.0.0/8      0.0.0.0/0
DROP      all  --  243.0.0.0/8      0.0.0.0/0
DROP      all  --  244.0.0.0/8      0.0.0.0/0
DROP      all  --  245.0.0.0/8      0.0.0.0/0
DROP      all  --  246.0.0.0/8      0.0.0.0/0
DROP      all  --  247.0.0.0/8      0.0.0.0/0
DROP      all  --  248.0.0.0/8      0.0.0.0/0
DROP      all  --  249.0.0.0/8      0.0.0.0/0
```

BIBLIOGRAPHY

1. Goncalves, Marcus. Firewall a Complete Guide. NY: McGraw-Hill, 2000.

2. Komar, Brain, Ronald Beekelaar, and Joern W. Ettern. Firewall for Dummies. CA: Hungry Minds Inc., 2001.

3. Ziegler, Robert L. Linux Firewall. Indiana: New Riders Publisher, 200.

4. Zwicky, Elizabeth, Simon Cooper, and Brent D. Chapman. Building Internet Firewalls. CA: O'Reilly and Associates, Inc., 2000.