بسم الله الرحمن الرحيم

Sudan University of Science & Technology

College of Postgraduate Studies

Design and Implementation of Internet Firewall

A thesis submitted in partial fulfillment of the requirements for the degree of M. Sc. in Information Technology

Prepared by:

Isam Abdel Nabi Osman Mohammed Ali

Supervised by:

Dr. Yahia Abdalla Mohammed

November 2005

بسم الله الرحمن الرحيم

قال تعالى :

 إقرأ باسم ربك الذي
خلق خلق الإنسان من
علق أقرأ وربك الأكرم
الذي علم بالقلم علم
الإنسان مالم يعلم

صدق الله العظيم

سورة العلق الايات 1-5

ACKNOWLEDGEMENT

I like to acknowledge the contributions of my supervisor Dr. Yahia Abdalla Mohammed who helps brilliant creativity support, and have guided this project from its conception to its completion, and I would like to send my special thanks to him.

I thank the teaching staff of the department of electronic engineering and International Institute of Information Technology (I^2 IT India), who gave me their considerations and were always behind me helping and encouraging.

Special thanks for Rehab Mohi eldeen, Hassabalah Abdel gayom, and all others for their usual supports without limitations, to accomplish this work and standing behind me.

DEDICATION

Thank you god, for giving me the courage, ability and strength to accomplish this work.

I dedicate this deepest love and affection to my parents. Their love, vulnerability and wisdom have inspired me to do the best I can.

To my friends, thanks for the endearing friendship, for the joy I have always shared, for the days filled with humor and laughter and for bond of endless understanding.

To all the people who helped me to accomplish this work.

With my greater love

Isam …

ABSTRACT

The need for security in the networks has become most essential due to many reasons, the most important reason is to protect data and information from hackers and keep confidential data out of reach from unauthorized access by other people.

One of the ways to implement security enforcement in network is to use *"Firewall Technique"*; Firewall technique is the process of separating protected network from unprotected network through central checkpoint with aid of software and hardware.

To maintain firewall, the network topology, cabling, system, devices, and OSI layers should be determined first. In addition to the above requirements it is important to select the protocol (TCP/IP), routing, and Client/Server interaction.

The firewall strategy is strongly based on security policy that has been selected and implemented by the firewall.

Firewalls has helped in improvement of security and reducing server risks and blocking the services that could attack by hackers, and provide us developing control access to network sites.

For importance of the firewalls this project discusses the principles of firewall, and the firewall types. A simple firewall has been designed to run under the Linux operating system; which consider as open source, inexpensive, and more

reliable than most desktop oriented operating systems. The firewall has been implemented, and gave a good result when tested.

مستخلص

الحاجة للامن فى الشبكات اصبحت من الضروريات لاسباب عديدة، أهم الأسباب التى تجعل تأمين الشبكات فى غاية الأهمية هو حماية المعلومات والبيانات من المتطفلين وايضا لحفظ البيانات السرية بعيدا عن وصول اى شخص غير مصرح له.

أحد أهم الطرق المستخدمة فى الحماية هى تقنية جدار النار وهى تقنية فصل الشبكة الداخلية المطلوب حمايتها عن الشبكات الخارجية فى نقطة مركزية يتركز فيها الأمن وذلك بواسطة برمجيات و عتاد.

نحتاج قبل تطبيق تقنية جدار النار فى أى شبكة لتحليل تلك الشبكة تحليلاً شاملاً موضحاً فيه التقنية الخاصة بالشبكة، والتشبيك والاجهزة والنظر للشبكة المحمية اعتمادا على طبقات نموذج الـ (OSI)، وكذلك من المهم معرفة نوع البروتوكول المستخدم والتوجيه وكذلك طبيعة عمل المستفيد/المخدم ووظيفة كل منهما.

ولتطبيق تقنية جدار النار لابد من تحديد المعايير الأمنية المطلوبة فى الشبكة فيما يدعى بالسياسة الامنية وتنفيذ هذه السياسة بواسطة جدار النار .

ساعدت جدران النار فى تطوير سرية الشبكات وتقليل المخاطر التى تتعرض لها المخدمات بالاضافة الى عزل الخدمات التى يمكن اسغلالها بواسطة المخربين وذلك بالتحكم فى الوصول الى الشبكة.

ولاهمية جدران النار تم فى هذا البحث التطرق الى اساسيات جدران النارواانواعها. تم تصميم جدار نار بسيط ليعمل على نظام التشغيل (Linux) والذى يمثل احد المصادر المفتوحة، الاقل تكلفة والاكثر اعتمادية مقارنة بانظمة التشغيل سطحية المكتب. تم تنفيذ جدار النار واعطى نتائج جيدة عند اختباره.

TABLE OF CONTENTS

1. LIST OF FIGURES

Chapter One

2. CHAPTER ONE: INTRODUCTION

2.1 Introduction

Computer networks have evolved considerably over the last 30 years. In the beginning, computer systems were stand-alone entities. Access to mainframes was provided via dumb terminals over serial line. All security and control was managed centrally. Early computer networks were usually comprised of private connections using propriety network media and protocols. Specific communication methods varied from vender to vender. This model started to change with the advent of local area networks (LANs) in the early 1980s and the introduction of Transmission Control Protocol/ Internet Protocol (TCP/IP). As the TCP/IP protocol gained momentum, distributed computing started to replace the mainframe/dumb terminal model. Interoperability between different platforms and standardization of network protocols stripped away the insulation of proprietary communication mechanisms. As this transformation took place, security challenges arose. However, TCP/IP and its network-based applications (Telnet, FTP, and HTTP) were not designed with security in mind. As Internet accessibility become a method of communication for commercial entities, security weakness were appeared. Companies were now able to directly communicate with their customers and business partners over low-cost public networks.

2.2 Problem Definition

As mentioned the Internet is a wonderful technological advance that provides access to information, and the ability to publish information, but it's also a major danger that provides the ability to infect and destroy information.

When a private network is attached to the Internet, there are three areas of potentials concerns or risk [1]:

- **Information**

  Someone can steal or destroy the information that stored on the private network.

- **Resources**

  Someone can damage or misuse the computer systems on the private network.

- **Reputation**

  Someone can damage the reputation of a business by demonstrating vulnerabilities in its network security.

So that this project is about one way to balance the advantages and the risks, to take part in the Internet while still protecting you using "*Firewalls*", which is a component or set of components that restricts access between a protected network and the Internet, or between other sets of networks.

2.3 Thesis Layout

Chapter One

This chapter gives introduction to *Firewall* and the risks when connecting a private network to the Internet.

Chapter Two

This chapter is about *Security policy*, and its contents and how to get Strategic and Policy Decisions Made

Chapter Three

This chapter is about a *Firewall architectures*, Dual-homed host architecture, Screened Host Architecture, and Screened Subnet Architecture with its components.

Chapter Four

This chapter is about the system environment, design, and implementation of the *Firewall.*

Chapter Five

This chapter is about the testing and result of the designed firewall with the conclusion and recommendations.

Chapter Two

3. CHAPTER TWO: BACKGROUND

3.1 Security Policies

A security policy is a formal statement of the rules that people who are given access to an organization's technology and information assets must follow it. The policy communicates the security goals to all of the users, the administrators, and the managers. The goals will be largely determined by the following key tradeoffs:

- Services offered versus security provided.
- Ease of use versus security.
- Cost of security versus risk of loss.

The main purpose of a security policy is to inform the users, the administrators and the managers of their obligatory requirements for protecting technology and information assets. The policy should specify the mechanisms through which these requirements can be met. Another purpose is to provide a baseline from which to acquire, configure and audit computer systems and networks for compliance with the policy. In order for a security policy to be appropriate and effective, it needs to have the acceptance and support of all levels of employees within the organization. A good security policy must [1]:

- Be able to be implemented through system administration procedures, publishing of acceptable use guidelines, or other appropriate methods
- Be able to be enforced with security tools, where appropriate, and with sanctions, where actual prevention is not technically feasible
- Clearly define the areas of responsibility for the users, the administrators, and the managers
- Be communicated to all once it is established
- Be flexible to the changing environment of a computer network since it is a living document

3.2 Your Security Policy

Most technical computer people consider a single, unified, published security policy to be desirable in the abstract, but attempting to come up with one is going to be extremely painful. There is no doubt that putting together a security policy is going to be a long, involved process, and that it's the exact opposite of the types of tasks most technical people enjoy.

Every one wants the best security that meets his requirements for [1]:

- **Affordability**

  How much money does the security cost?

- **Functionality**

  Can you still use your computers?

- **Cultural compatibility**

  Does it conflict with the way people at your site normally interact with each other and the outside world?

- **Legality**

  Does it meet the site's legal requirements?

3.3 Security Policy Contents

First and foremost, a security policy is a way of communicating with users and managers. It should tell them how to make their security decisions.

3.3.1 Explanations

It's important that the policy be explicit and understandable about why certain decisions have been made. Most people will not follow rules unless they understand why they are important. A policy that specifies what's supposed to be done, but not why, is doomed. As soon as the people who wrote it leave, or forget why they made those decisions, it's going to stop having any effect.

3.3.2 Everybody's responsibilities

A policy sets explicit expectations and responsibilities users, and mangier; it lets all of you know what to expect from each other. It's a mistake

to distribute a policy that concentrates entirely on what users need to do to make the site secure, or entirely on what system administrators need to do.

3.3.3 Regular language

It's more important to make your security policy friendly and understandable than to make it precise and official-looking. Write it as if you were explaining it to a reasonably bright but non technical friend. Keep it a communication between peers. If that's not acceptable in your corporate culture, write two separate policy descriptions.

3.3.4 Enforcement authority

Writing down the policy is not the point; living by it is. That means that when the policy isn't followed, something should happen to fix it. Somebody needs to be responsible for making those corrections happen, and the policy needs to specify how that's going to be and the general range of corrections. Here are some examples of what a security policy might specify:

- Managers of certain services have the authority to revoke access.
- Managers will be asked to take care of some kinds of transgressions.
- Facilities that don't meet certain standards may be cut off from the corporate network and external access by the people who it.

The policy should specify who is going to decide and give some indications of what kinds of penalties are available to them. It should not specify exactly what will happen when; it's a policy, not a mandatory sentencing law.

3.3.5 Provision for reviews

It can't expect to set a policy up once and forget it. The needs of your site will change over time, and policies that were perfectly sensible may become either too restrictive or too lax. You still need to review and change your policies on a regular basis.

3.4 Getting Strategic Decisions made

Strategic decisions need to be understood and made by top-level management or they will never be successfully implemented. If the top-level management dose not support for security, there is no way to have security; it's that simple. Here are some things to consider in making decisions.

3.4.1 Insertion of Everybody Who's Affected

Strategic and policy decisions must be made by people working together. No body can't just come up with a policy he likes, take it around to a lot of people, and have them rubber stamp it. Even if he manages to get them to do it which may well be more difficult than getting them to help make intelligent decisions they won't actually follow it.

3.4.2 Wrong Decisions Acceptability

Sometimes the security policy we come up with is one that we don't particularly like. If this happens because the people who made it don't understand what they have done, then the fight should be done strongly to get it fixed. If, on the other hand, people understand the risks, decision like this one may be acceptable.

3.4.3 Surprises Avoidance

When it comes to security, nobody likes surprises. That's why the need for making sure that the relevant people understand the relevant issues and are aware of, and agree with, the decisions made concerning those issues. In particular, people need to know about the consequences of their decisions, including best, worst, and probable outcomes. Consequences that are obvious to someone may not be obvious to other people.

3.4.4 Concentration to the Important Decisions, with Implications

The decision shouldn't offer to the people unless they have both the authority and the information with which to make those decisions. Always

make it clear why they are being asked to decide. In most cases, the avoidance of open-ended questions is needed, because it gives the replier the option of saying "nothing," which is probably not an answer for making decisions.

3.4.5 Justification of Everything Else in Terms of Those Decisions

All of the technical and implementation decisions make should follow from the high-level guidance obtained from the top managers and executives. If there is no way to go with a technical issue because it depends on non technical issues, there is a need to request more guidance on that issue. The explanation of the problem must be clearly; the options and the implications of each option.

3.4.6 Indication to the NonTechnical Issues

Certain problems, which some people try to characterize or solve as technical problems, are really management or personnel problems. For example, some managers worry that their employees will spend all their time at work reading use net news or surfing the Web. However, this is not a technical problem, but a personnel problem: the online equivalent of employees spending the day at their desks reading the newspaper.

3.4.7 Understanding of Anything

Certain things that seem obvious to a technical person who is interested in security may not be at all obvious to non technical managers and executives. For example it's obvious to anyone who understands IP that packet filtering will allow you to restrict access to services by IP addresses, but not by user (unless you can tie specific users to specific IP addresses), because "user" is not a concept in IP, and there's nothing in the IP packet that reflects what "user" is responsible for that packet. Conversely, certain things that seem obvious to managers and executives are not at all obvious to technical staff.

Chapter Three

4. CHAPTER THREE: FIREWALLS

4.1 Internet Firewall

The Internet is a wonderful 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 infect and destroy information in revolutionary ways. One way to balance the advantages and the risks, to take part in the Internet while still protecting sites is by "*Firewalls*".

A firewall is a component or set of components that restricts access between a protected network and the Internet, or between other sets of networks.

This section briefly describes what Internet firewalls can do for your overall site security.

In building construction, a firewall is designed to keep a fire from spreading from one part of the building to another. In theory, an Internet firewall serves a similar purpose: it prevents the dangers of the Internet from spreading to your internal network. In practice, an Internet firewall is more like a moat of a medieval castle than a firewall in a modern building. It serves multiple purposes [1]:

- It restricts entrance of people at a carefully controlled point.
- It prevents attackers from getting close to your other defenses.
- It restricts people to leaving at a carefully controlled point.

Logically, a firewall is a separator, a restrictor, an analyzer. Most often, a firewall is a set of hardware components - a router, a host computer, or some combination of routers, computers, and networks with appropriate software. An Internet firewall is most often installed at the point where your protected internal network connects to the Internet, as shown in Figure 3.1.

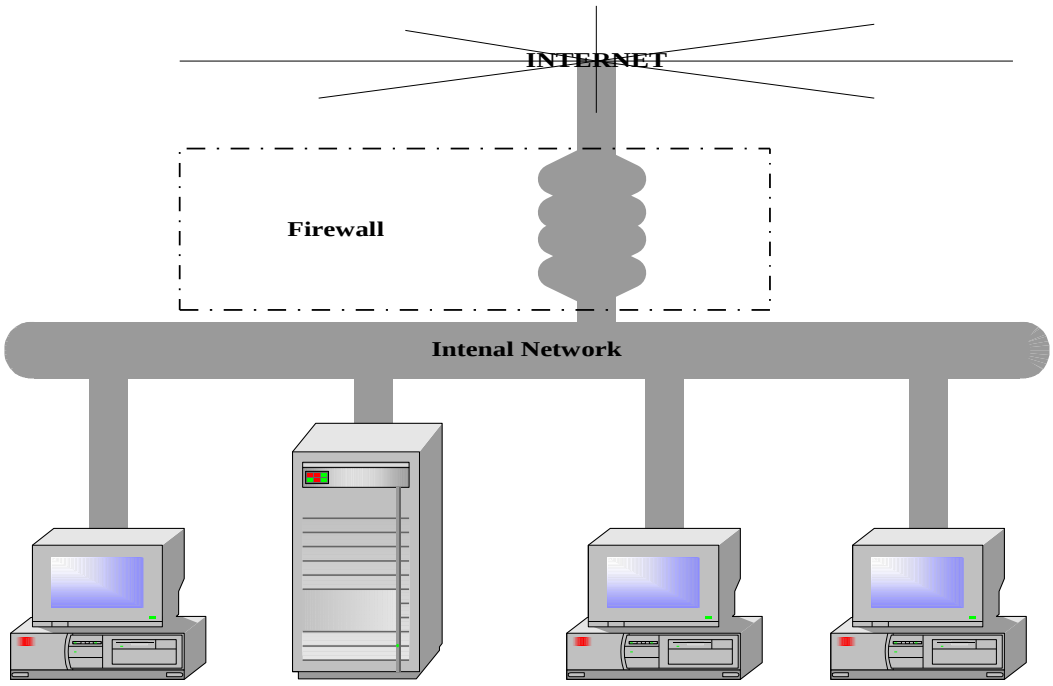


Figure 3.1: A firewall usually separates an internal network from the Internet

All traffic coming from the Internet or going out from your internal network passes through the firewall. Because it does, the firewall has the opportunity to make sure that this traffic is acceptable, which means that whatever is being done - email, file transfers, remote logins, or any kinds of specific interactions between specific systems - conforms to the security policy of the site. Security policies are different for every site; some are highly restrictive and others fairly open. The physical implementation of the firewall may vary from site to site. There are various ways to configure this equipment; the configuration will depend upon a site's particular security policy, budget, and overall operations. A firewall is very rarely a single physical object, although some of the newest commercial products attempt to put everything into the same box. Usually, a firewall has multiple parts, and some of these parts may do other tasks besides functioning as part of the firewall. Your Internet connection is almost always part of your firewall. Even if you have a firewall in a box, it isn't going to be neatly separable from the rest of your site; it's not something you can just drop in. Given the limitations and drawbacks of firewalls, why would anybody bother to install one? Because a firewall is the most effective way to connect a network to the Internet and still protect that network. The Internet presents wonderful opportunities. Millions of people are out there exchanging information. The benefits are obvious: the chances for publicity,

customer service, and information gathering. The popularity of the information superhighway is increasing everybody's desire to get out there. The risks should also be obvious: any time you get millions of people together, you get crime; it's true in a city, and it's true on the Internet. Any superhighway is fun only while you are in a car. If you have to live or work by the highway, it's loud, smelly, and dangerous. How can you benefit from the good parts of the Internet without being overwhelmed by the bad? You need to carefully control the contact that your network has to the Internet. A firewall is a tool for doing that, and in most situations, it's the single most effective tool for doing that. There are other uses of firewalls. For example, they can be used as firewalls in a building that divide parts of a site from each other when these parts have distinct security needs. Firewalls offer significant benefits, but they can't solve every security problem. Firewalls can do a lot for your site's security as described below [1].

- **A firewall is a focus for security decisions**

  Think of a firewall as a choke point. All traffic in and out must pass through this single, narrow checkpoint. A firewall gives you an enormous amount of leverage for network security because it lets you concentrate your security measures on this checkpoint: the point where your network connects to the Internet.

- **A firewall can enforce security policy**

  Many of the services that people want from the Internet are inherently insecure. The firewall is the traffic cop for these services. It enforces the site's security policy, allowing only "approved" services to pass through and those only within the rules set up for them.

- **A firewall can log Internet activity efficiently**

  Because all traffic passes through the firewall, the firewall provides a good place to collect information about system and network use - and misuse. As a single point of access, the firewall can record what occurs between the protected network and the external network. Firewalls offer excellent protection against network threats, but they aren't a complete security solution. Certain threats are outside the control of the firewall. You need to figure out other ways to protect against these threats by incorporating

physical security, host security, and user education into your overall security plan. Some of the weaknesses of firewalls are discussed below.

- **A firewall can't protect you against malicious insiders**

A firewall might keep a system user from being able to send proprietary information out of an organization over a network connection; so would simply not have a network connection. But that same user could copy the data onto disk, tape, or paper and carry it out of the building in his or her briefcase.

- **firewall can't protect against connections don't go through it**

A firewall can effectively control the traffic that passes through it; however, there is nothing a firewall can do about traffic that doesn't pass through it

- **A firewall can't protect against viruses**

Firewalls can't keep PC viruses out of a network. Although many firewalls scan all incoming traffic to determine whether it is allowed to pass through to the internal network, the scanning is mostly for source and destination addresses and port numbers, not for the details of the data. Even with sophisticated packet filtering or proxying software, virus protection in a firewall is not very practical. There are simply too many types of viruses and too many ways a virus can hide within data.

4.2 Firewall Architectures

There are three types of firewall architecture, which will be described in details in the next section [1]:

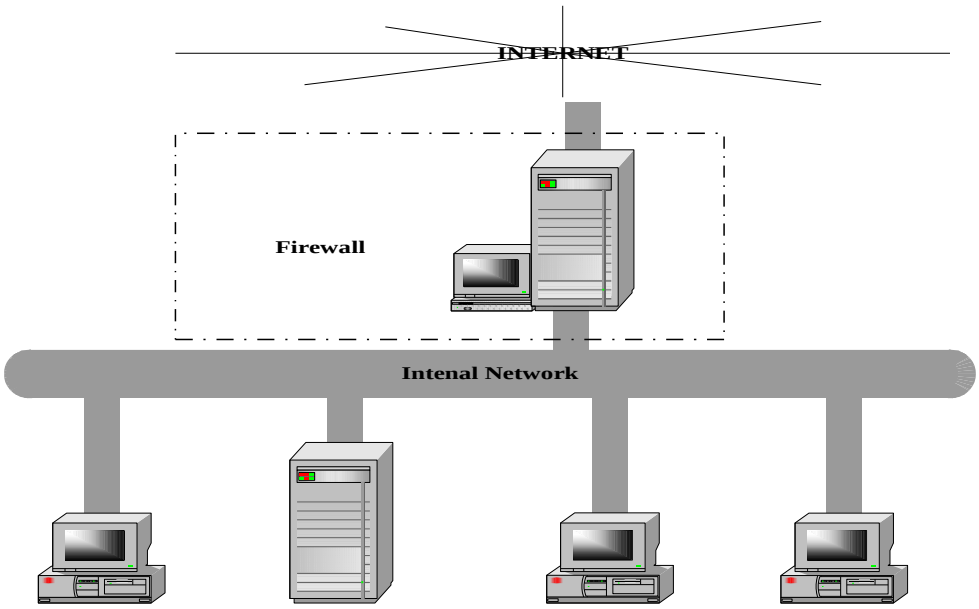
- *Dual-Homed Host Architecture*
- *Screened Host Architecture*
- *Screened Subnet Architecture*

4.2.1 Dual-Homed Host Architecture

Dual-homed host architecture is built around the dual-homed host computer, a computer which has at least two network interfaces. Such a host could act as a router between the networks these interfaces are attached to; it is capable of routing IP packets from one network to another. However, to

implement a dual-homed host type of firewalls architecture, you disable this routing function. Thus, IP packets from one network (e.g., the Internet) are not directly routed to the other network (e.g., the internal, protected network). Systems inside the firewall can communicate with the dual-homed host, and systems outside the firewall (on the Internet) can communicate with the dual-homed host, but these systems can't communicate directly with each other. IP traffic between them is completely blocked. The network architecture for a dual-homed host firewall is pretty simple: the dual homed host sits between, and is connected to, the Internet and the internal network. Figure 3.2 shows this architecture.

Figure 3.2: Dual-homed host architecture



Dual-homed hosts can provide a very high level of control. If the packets aren't allowing between external and internal networks at all, you can be sure that any packet on the internal network that has an external source is evidence of some kind of security problem. In some cases, a dual-homed host will allow you to reject connections that claim to be for a particular service but that didn't actually contain the right kind of data. A packet filtering system, on the other hand, has difficulty with this level of control; however, it takes considerable work to consistently take advantage of the potential advantages of dual-homed hosts. A dual-homed host can only provide services by proxying them, or by having users log into the

dual-homed host directly, user accounts present significant security problems by themselves. They present special problems on dual-homed hosts, where they may unexpectedly enable services you consider insecure. Furthermore, most users find it inconvenient to use a dual-homed host by logging into it. Proxying is much less problematic, but may not be available for all services you are interested in.

4.2.2 Screened Host Architecture

Whereas a dual-homed host architecture provides services from a host that's attached to multiple networks (but has routing turned off), a screened host architecture provides services from a host that's attached to only the internal network, using a separate router. In this architecture, the primary security is provided by packet filtering. (For example, packet filtering is what Prevents people from going around proxy servers to make direct connections.) Figure 3.3 shows a simple version of screened host architecture.

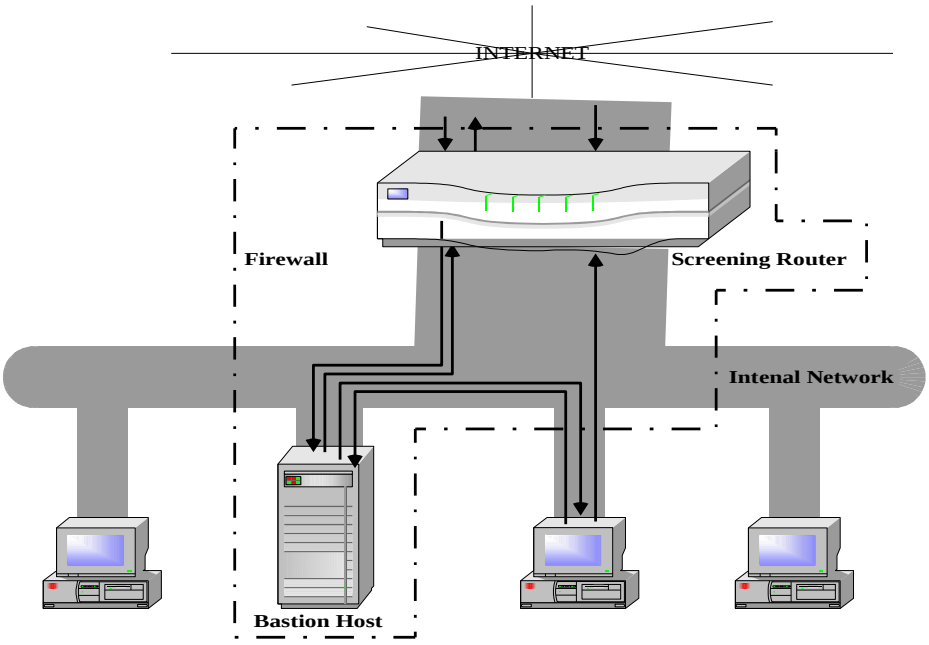


Figure 3.3: Screened host architecture

The bastion host sits on the internal network. The packet filtering on the screening router is set up in such a way that the bastion host is the only system on the internal network that hosts on the Internet can open

connections to (for example, to deliver incoming email). Even then, only certain types of connections are allowed. Any external system trying to access internal systems or services will have to connect to this host. The bastion host thus needs to maintain a high level of host security. The packet filtering also permits the bastion host to open allowable connections (what is "allowable" will be determined by your site's particular security policy) to the outside world. The packet filtering configuration in the screening router may do one of the following:

- Allow other internal hosts to open connections to hosts on the Internet for certain services (allowing those services via packet filtering).
- Disallow all connections from internal hosts (forcing those hosts to use proxy services via the bastion host.

You can mix and match these approaches for different services; some may be allowed directly via packet filtering, while others may be allowed only indirectly via proxy. It all depends on the particular policy your site is trying to enforce. Because this architecture allows packets to move from the Internet to the internal networks, it may seem more risky than a dual-homed host architecture, which is designed so that no external packet can reach the internal network. In practice, however, the dual-homed host architecture is also prone to failures that let packets actually cross from the external network to the internal network. (Because this type of failure is completely unexpected, there are unlikely to be protections against attacks of this kind.) Furthermore, it's easier to protect a router, which provides a very limited set of services, than it is to protect a host. For most purposes, the screened host architecture provides both better security and better usability than the dual-homed host architecture.

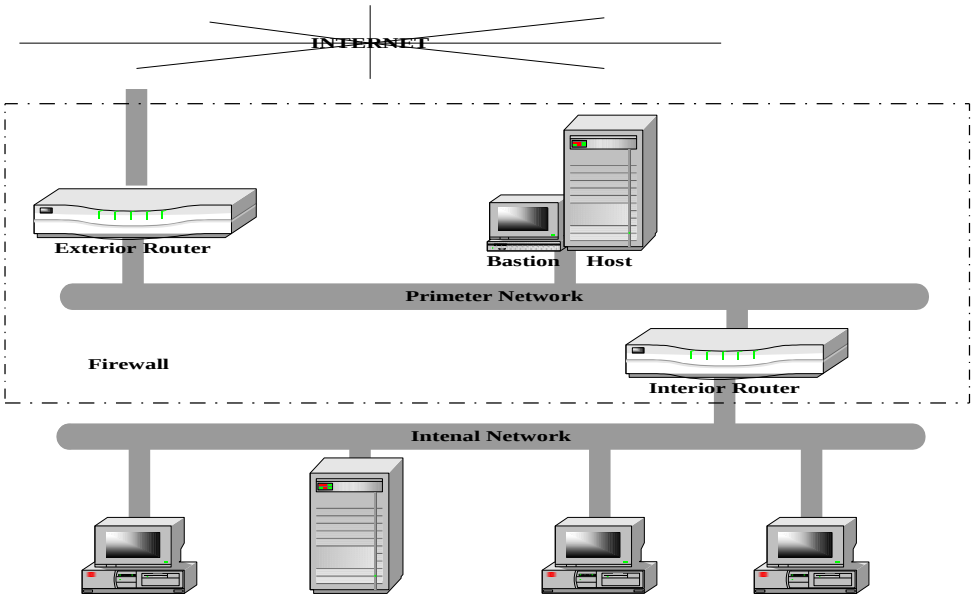
4.2.3 Screened Subnet Architecture

The screened subnet architecture adds an extra layer of security to the screened host architecture by adding a perimeter network that further isolates the internal network from the Internet, by their nature, bastion hosts are the most vulnerable machines on your network. Despite your best efforts to protect them, they are the machines most likely to be attacked, because they are the machines that can be attacked. If, as in screened host architecture,

your internal network is wide open to attack from your bastion host, then your bastion host is a very tempting target. There are no other defenses between it and your other internal. If someone successfully breaks into the bastion host in screened host architecture, he's hit the jackpot. By isolating the bastion host on a perimeter network, you can reduce the impact of a break-in on the bastion host. It is no longer an instantaneous jackpot; it gives an intruder some access, but not all. With the simplest type of screened subnet architecture, there are two screening routers, each connected to the perimeter net. One sits between the perimeter net and the internal network, and the other sits between the perimeter net and the external network (usually the Internet). To break into the internal network with this type of architecture, an attacker would have to get past both routers. Even if the attacker somehow broke in to the bastion host, he should still have to get past the interior router. There is no single vulnerable point that will compromise the internal network. Figure 3.4 shows a possible firewall configuration that uses the screened subnet architecture. The next few sections describe the components in this type of architecture.

Figure 3.4: Screened subnet architecture (using two routers)

4.2.3.1 Perimeter network



The perimeter network is another layer of security, an additional network between the external network and your protected internal

network. If an attacker successfully breaks into the outer reaches of your firewall, the perimeter net offers an additional layer of protection between that attacker and your internal systems. Here's an example of why a perimeter network can be helpful. In many network setups, it's possible for any machine on a given network to see the traffic for every machine on that network. This is true for most Ethernet-based networks, (and Ethernet is by far the most common local area networking technology in use today); it is also true for several other popular technologies, such as token ring and FDDI. Snoopers may succeed in picking up passwords by watching for those used during Telnet, FTP, and rlogin sessions. Even if passwords aren't compromised, snoopers can still peek at the contents of sensitive files people may be accessing, interesting email they may be reading, and so on; the snooper can essentially "watch over the shoulder" of anyone using the network. With a perimeter network, if someone breaks into a bastion host on the perimeter net, he shall be able to snoop only on traffic on that net. All the traffic on the perimeter net should be either to or from the bastion host, or to or from the Internet. Because no strictly internal traffic (that is, traffic between two internal hosts, which is presumably sensitive or proprietary) passes over the perimeter net, internal traffic will be safe from prying eyes if the bastion host is compromised. Obviously, traffic to and from the bastion host, or the external world, will still be visible

4.2.3.2 Bastion host

Bastion Host is a computer system that must be highly secured because it is exposed to the Internet and is a main point of contact for users of internal network. With the screened subnet architecture, you attach a bastion host (or hosts) to the perimeter net; this host is the main point of contact for incoming connections from the outside world; for example:

- For incoming email (SMTP) sessions to deliver electronic mail to the site

- For incoming FTP connections to the site's anonymous FTP server
- For incoming domain name service (DNS) queries about the site.

Outbound services (from internal clients to servers on the Internet) are handled in either of these ways:

Set up packet filtering on both the exterior and interior routers to allow internal clients to access external servers directly.

Set up proxy servers to run on the bastion host (if your firewall uses proxy software) to allow internal clients to access external servers indirectly. You would also set up packet filtering to allow the internal clients to talk to the proxy servers on the bastion host and vice versa, but to prohibit direct communications between internal clients and the outside world. In either case, the packet filtering allows the bastion host to connect to, and accept connections from, hosts on the Internet; which hosts, and for what services, are dictated by the site's security policy.

4.2.3.3 Interior router

The interior router (sometimes called the choke router in firewalls literature) protects the internal network both from the Internet and from the perimeter net. The interior router does most of the packet filtering for your firewall. It allows selected services outbound from the internal net to the Internet. These services are the services your site can safely support and safely provide using packet filtering rather than proxies. The services you allow might include outgoing Telnet, FTP, WAIS, Archie, Gopher, and others, as appropriate for your own needs and concerns. The services the interior router allows between your bastion host (on the perimeter net itself) and your internal net are not necessarily the same services the interior router allows between the Internet and your internal net. The reason for limiting the services between the bastion host and the internal network is to reduce the number of machines (and the number of services on those machines) that can be attacked from the bastion host, should it be compromised.

4.2.3.4 Exterior router

In theory, the exterior router (sometimes called the access router in firewalls literature) protects both the perimeter net and the internal net from the Internet. In practice, exterior routers tend to allow almost anything outbound from the perimeter net, and they generally do very little packet filtering. The packet filtering rules to protect internal machines would need to be essentially the same on both the interior router and the exterior router; if there's an error in the rules that allows access to an attacker, the error will probably be present on both routers. Frequently, the exterior router is provided by an external group, and your access to it may be limited. An external group that's maintaining a router will probably be willing to put in a few general packet filtering rules, but wouldn't want to maintain a complicated or frequently changing rule set. You also may not trust them as much as you trust your own routers. If the router breaks and they install a new one, are they going to remember to reinstall the filters? Are they even going to bother to mention that they replaced the router so that you know to check? The only packet filtering rules that are really special on the exterior router are those that protect the machines on the perimeter net (that is, the bastion hosts and the internal router). Generally, however, not much protection is necessary, because the hosts on the perimeter net are protected primarily through host security. The rest of the rules that you could put on the exterior router are duplicates of the rules on the interior router. These are the rules that prevent insecure traffic from going between internal hosts and the Internet. To support proxy services, where the interior router will let the internal hosts send some protocols as long as they are talking to the bastion host, the exterior router could let those protocols through as long as they are coming from the bastion host. These rules are desirable for an extra level of security, but they are theoretically blocking only packets that can't exist because they have already been blocked by the interior router. If they do exist, either the interior router has failed, or somebody has connected an unexpected host to the perimeter network. So, what does the exterior router actually need

to do? One of the security tasks that the exterior router can usefully perform - a task that usually can't easily be done anywhere else - is the blocking of any incoming packets from the Internet that have forged source addresses. Such packets claim to have come from within the internal network, but actually are coming in from the Internet. The interior router could do this, but it can't tell if packets that claim to be from the perimeter net are forged. While the perimeter net shouldn't have anything fully trusted on it, it's still going to be more trusted than the external universe; being able to forge packets from it will give an attacker most of the benefits of compromising the bastion host. The exterior router is at a clearer boundary. The interior router also can't protect the systems on the perimeter net against forged packets.

Chapter Four

5. CHAPTER FOUR: SYSTEM ENVIRONMENT

5.1 Introduction

The environment of the designed firewall consist of Linux Red Hat 8 as an Operating System, and Gimp Tool Kit (GTK) as a tool to create the graphical user interface of the simple firewall (software components), and general purpose personal computer with two network cards, and network switches (hardware components).

5.2 Linux Operating System

Linux (pronounced *Lih-nucks*) is a UNIX-like operating system that runs on many different computers. Although many people might refer to Linux as the operating system and included software, strictly speaking, Linux is the operating system *kernel*, which comes with a *distribution* of software. Linux was first released in 1991 by its author *Linus Torvalds* at the University of Helsinki. Since then it has grown tremendously in popularity as programmers around the world embraced his project of building a free operating system, adding features, and fixing problems. Linux is popular with today's generation of computer users for the same reasons early versions of the UNIX operating system enticed fans more than 20 years ago. Linux is portable, which means you'll find versions running on name-brand or clone PCs, Apple Macintoshes, Sun workstations, or Digital Equipment Corporation Alpha-based computers. Linux also comes with source code, so you can change or customize the software to adapt to your needs. Red Hat Linux is the most popular commercial distribution of Linux. Red Hat and other commercial distributions, such as Caldera's, OpenLinux have taken the Linux concept a step further. With Red Hat Linux users can rely on Red Hat Software to provide tested versions of that software and technical support if there are problems. Finally, Linux is a great operating system, rich in features adopted from other versions of UNIX [2]:

5.2.1 Primary Advantages of Linux

When compared to different commercially available operating systems, Linux is best assets are its price and its reliability. In terms of reliability, the general agreement is that Linux is comparable to many commercial UNIX systems but more reliable than most desktop-oriented operating systems. This is especially true if you rely on your computer system to stay up because it is a Web server or a file server. Another advantage of using Linux is that help is always available on the Internet.

5.2.2 Common Linux Features

No matter what version of Linux you use, the price of code common to all is Linux kernel. Although the kernel can be modified to include support for features you want, every Linux kernel can offer the following features:

- **Multi-user**

  Not only can you have many user accounts available on Linux system, you can also have multiple users logged in and working on the system at the same time. User can have their own environments arranged the way the want.

- **Multitasking**

  In Linux, it is possible to have many programs running at the same time, which means that not only can you have many programs going at once, but that the Linux Operating System can itself have programs running in the background. Many of these system processes make it possible for Linux to work as a server, with these background processes listening to the network for requests to log in to your system.

- **Graphical User Interface (X Widow System)**

  The powerful framework for working with graphical applications in Linux is referred to as ***X Widow System***, which handles the functions of opening X-based GUI applications and displaying them on an X server process.

- **Hardware Support**

You can configure support for almost every type of hardware that can be connected to a computer. There is support for floppy disk drive, CD-ROMs, removable disks, sound cards and so on…

- **Networking Connectivity**

To connect your Linux system to a network, Linux offers support a variety of local area network (LAN) boards, modems, and serial devices. The most popular protocol supported by the Linux is TCP/IP (which used to connect to the Internet). Other protocols, such as IPX (for Novell networks), and X.25 (a packet-switching network type that is popular in Europe), are available.

- **Network Servers**

Providing networking services to the client computers on LAN or to the entire Internet is what Linux dose the best. A variety of software packages are available that enable you to use Linux as a print server, file server, FTP server, mail server, web server etc.

- **Application Support**

Because of compatibility with POSIX and several different application programming interface (APIs); a wide range of freeware and shareware software is available for Linux. The most important thing for us here is that Linux has more than one tool to create GUI applications like GTK (Gimp Tool Kit) which used to create the code with aid of standard C-language.

5.2.3 Chosen Red Hat Linux 8

To distinguish themselves from other versions of Linux, each distribution adds some extra features so the Red Hat Linux 8 has been chosen for following reasons which are:

- Desktop environment system.
- Easy installation.
- Administration tools.
- Easy to configure and play with.
- Have more built in tools to create GUI applications like GTK.

5.3 Gimp Tool Kit (GTK)

The Gimp Tool Kit (GTK) is widely used for writing X Windows applications on Linux and other versions of UNIX. In order to help maintain both portability and software maintenance, GTK is built on top of two other libraries that you may want to use independently of GTK:

- **Glib:**

  Supplies C libraries for linked lists, hash tables, string utilities, and so on.

- **GDK:**

  A library that is layered on Xlib. All GTK windowing and graphics calls are made through GDK, not directly to XLib.

GTK has its own Web site (www.gtk.org) where latest version of GTK is available and read a very good GTK tutorial. In this part, we will introduce GTK. The GTK is an easy-to-use, high-level toolkit for writing GUI applications. GTK was written to support the GIMP graphics-editing program. GTK was originally written by *Peter Mattis*, *Spencer Kimball*, and *Josh MacDonald*. The GTK toolkit contains a rich set of data types and utility functions. A complete description of GTK is beyond the scope of this short introductory part. Still, reading this chapter will get you started. For more information, consult the online tutorial and example programs included in the examples directory in the standard GTK distribution [9].

GTK has its own type of widget; the C data structure for this type is called GtkWidget Windows and any display components created using GTK can all be referenced using a variable of type GtkWidget. Although GTK is written in C, GTK has a strong object-oriented flavor. A GtkWidget object encapsulates data, maintains references to Callback functions, and so on.

GTK widgets “inherit” from GtkWidget by defining a GtkWidget object as the first item in their own structure definition. A pointer to any type of GtkWidget can be safely coerced to the type (GtkWidget *) and the fields in the GtkWidget structure can be directly accessed. So, even though GTK is implemented in the C programming language and does not support private data, it has an object-oriented flavor [9].

5.4 Design of the environment

The design of the network used to implement the firewall on it is assumed to publish website via Internet connection and give an FTP service. It has only one IP address, and the HTTP and FTP servers are located on the internal network. The firewall has the external IP address INET_IP (212.1.1.1), and the HTTP server has the internal IP address HTTP_IP (192.168.2.28) and the FTP server has the internal IP address FTP_IP (192.168.2.30) and finally the firewall has the internal IP address LAN_IP (192.168.2.29) as shown in figures 4.1, and 4.2 below.

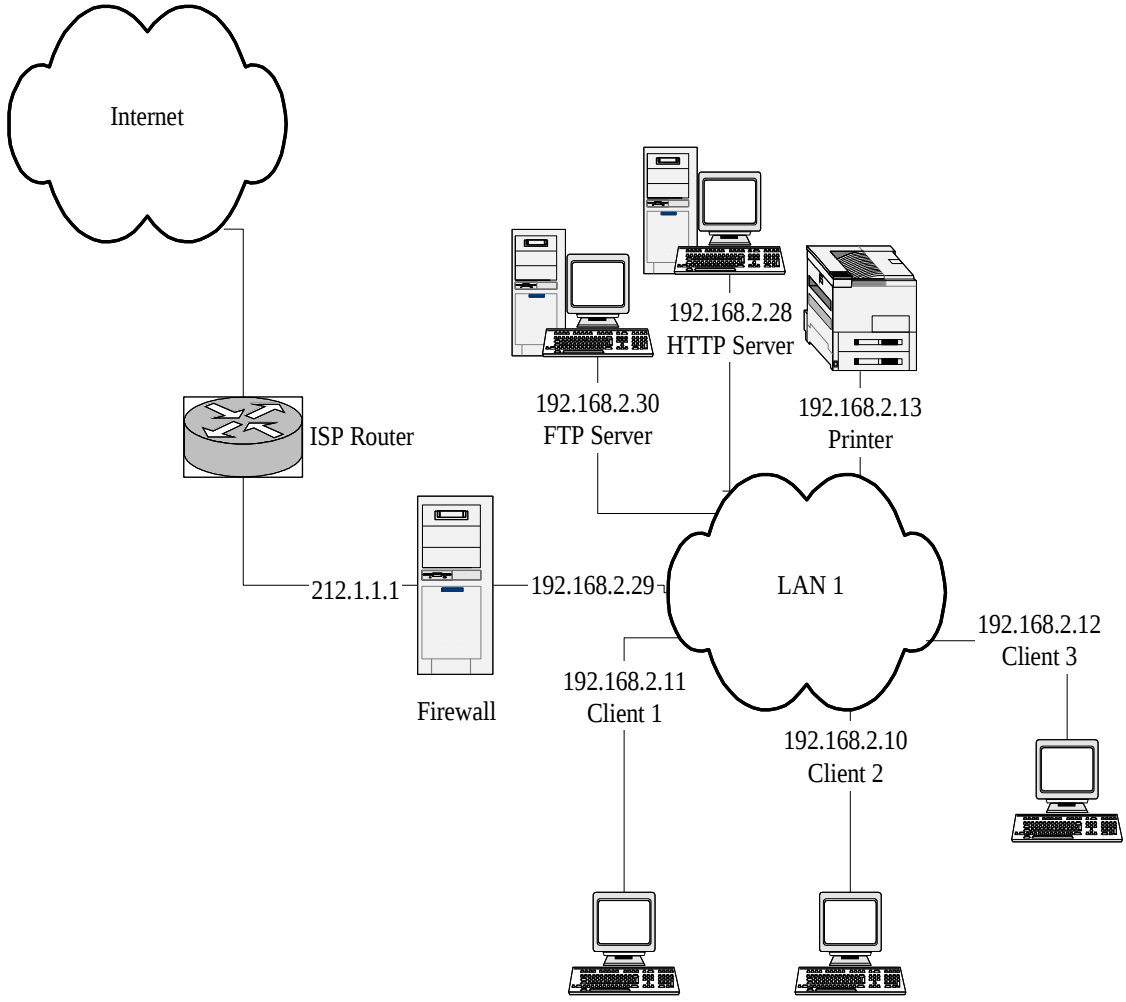


Figure 4.1: Logical Network Design

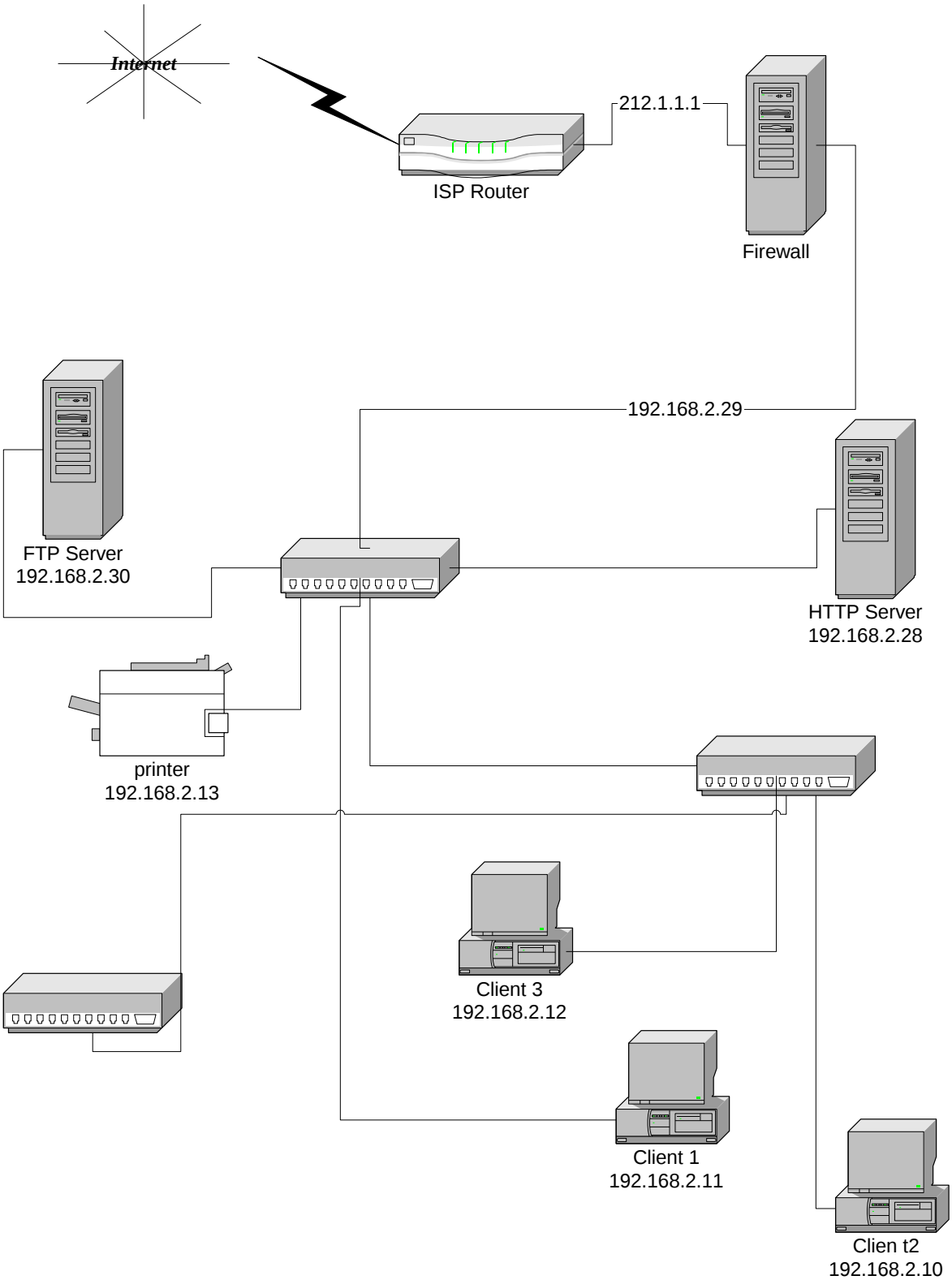


Figure 4.2: Physical Network Design

5.5 Implementation

All packets from the Internet going to port 80 on our firewall with INET_IP are redirected to our internal HTTP server and all packets from the Internet going to port 21 on our firewall with INET_IP are redirected to our internal FTP server after passing the incoming rule table file and check all lines on it for legality of the packet to pass. First it check the destination address, destination port, protocol type, and source address with that extract from the incoming packet and compare all with each line if they are equally with one line it apply the policy specified in that line (accept, reject, or drop) . All packets from the local LAN must change their source address to be INET_IP after passing the outgoing rule table file and check all lines on it for legality of the packet to pass first it check the destination address, protocol type, and source address with that extract from the outgoing packet and compare all with each line if they are equally with one line it apply the policy specified in that line (accept, reject, or drop). Implementing of firewall happen with those scenarios:

5.5.1 External computer want connect to our HTTP server:

The external box has IP address EXT_BOX, to maintain readability.

Packet leaves the connecting host going to INET_IP and source EXT_BOX.

- Packet reaches the firewall.
- Firewall check the port is it 80.
- Firewall check is it an illegal packet or not at the incoming table?
- If it’s not legal it will drop it or rejected according to the rule.
- If it is legal firewall redirect the packet to the HTTP server.
- Packet leaves the firewall and travels to the HTTP_IP.
- Packet reaches the HTTP server, and the HTTP box reply back through the firewall, if that is the box that the routing database has entered as the gateway for EXT_BOX. Normally, this would be the default gateway of the HTTP server.
- Firewall redirects the packet again, so the packet looks as if it was replied to from the firewall itself.
- Reply packet travels as usual back to the client EXT_BOX.

5.5.2 External computer want connect to our FTP server:

Packet leaves the connecting host going to INET_IP and source EXT_BOX.

- Packet reaches the firewall.
- Firewall check the port is it 21.
- Firewall check is it an illegal packet or not at the incoming table?
- If it's not legal it will drop it or rejected according to the rule.
- If it is legal firewall redirect the packet to the FTP server.
- Packet leaves the firewall and travels to the FTP_IP.
- Packet reaches the FTP server, and the FTP box reply back through the firewall, if that is the box that the routing database has entered as the gateway for EXT_BOX. Normally, this would be the default gateway of the FTP server.
- Firewall redirects the packet again, so the packet looks as if it was replied to from the firewall itself.
- Reply packet travels as usual back to the client EXT_BOX.

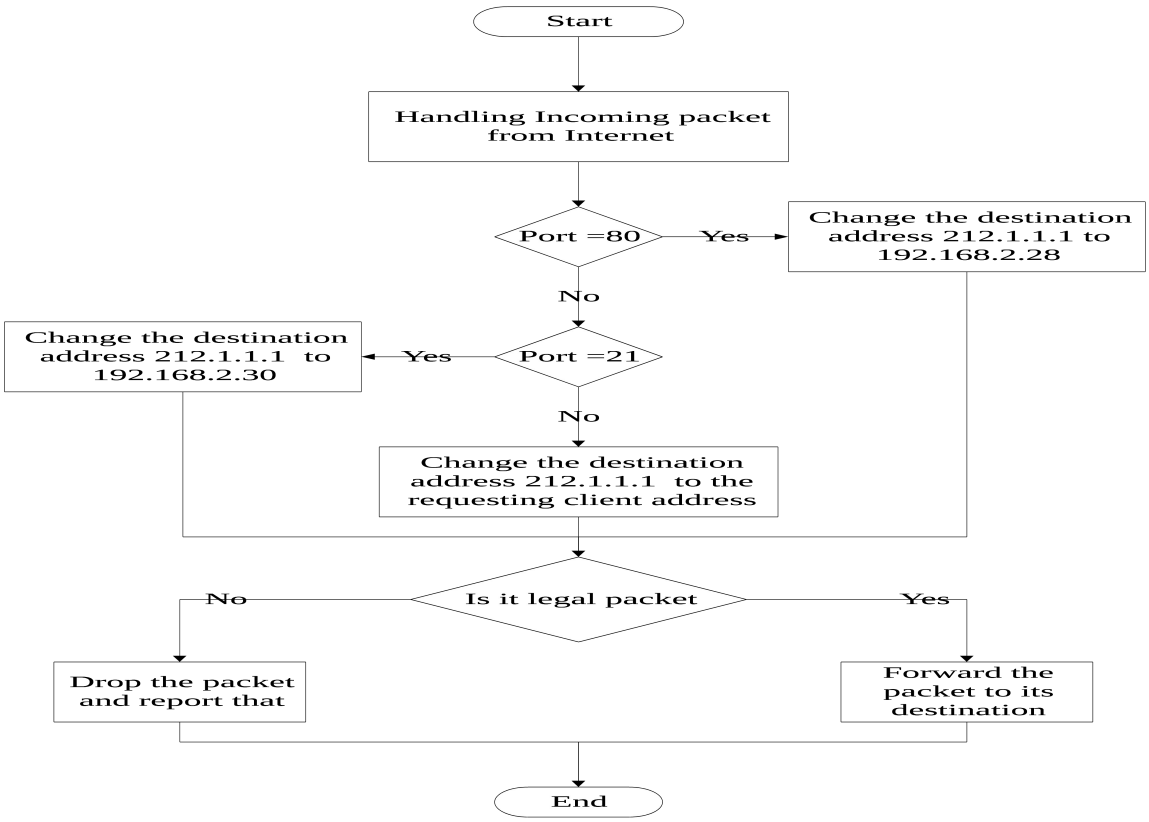


Figure 4.3: Flowchart for the packet from internet to our HTTP/FTP Server

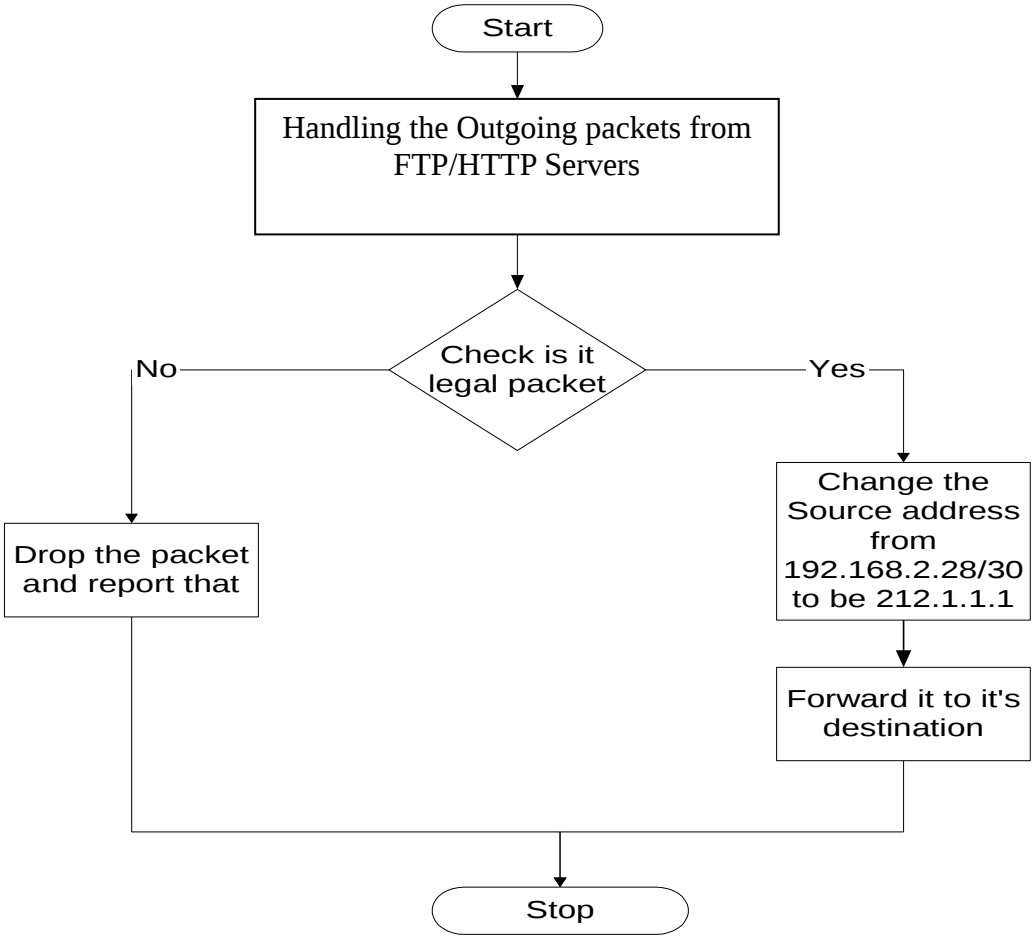


Figure 4.4: Flowchart for the packet from our HTTP/FTP Servers to Internet

5.5.3 Internal computer want connect to our FTP /HTTP server:

Now, we will consider what happens if the packet was instead generated by a client on the same network as the HTTP or FTP server itself. The client has the IP address **LAN_BOX**, while the rest of the machines maintain the same settings.

- Packet leaves **LAN_BOX** to **INET_IP**.
- The packet reaches the firewall.
- Firewall check the port is it 80 to redirect the packet the HTTP server or FTP server the port is 21.
- Firewall check is it an illegal packet or not at the outgoing table?
- If it’s not legal it will drop it or rejected according to the rule.

- The packet gets changing in source IP address to be the internal firewall IP address.
- The packet leaves the firewall and reaches the HTTP or FTP server. Packet reaches the HTTP or FTP server, and the HTTP or FTP box replies back through the firewall.
- Firewall redirects the packet again, so the packet looks as if it was replied to from the firewall itself.
- Reply packet travels as usual back to the client LAN_BOX.

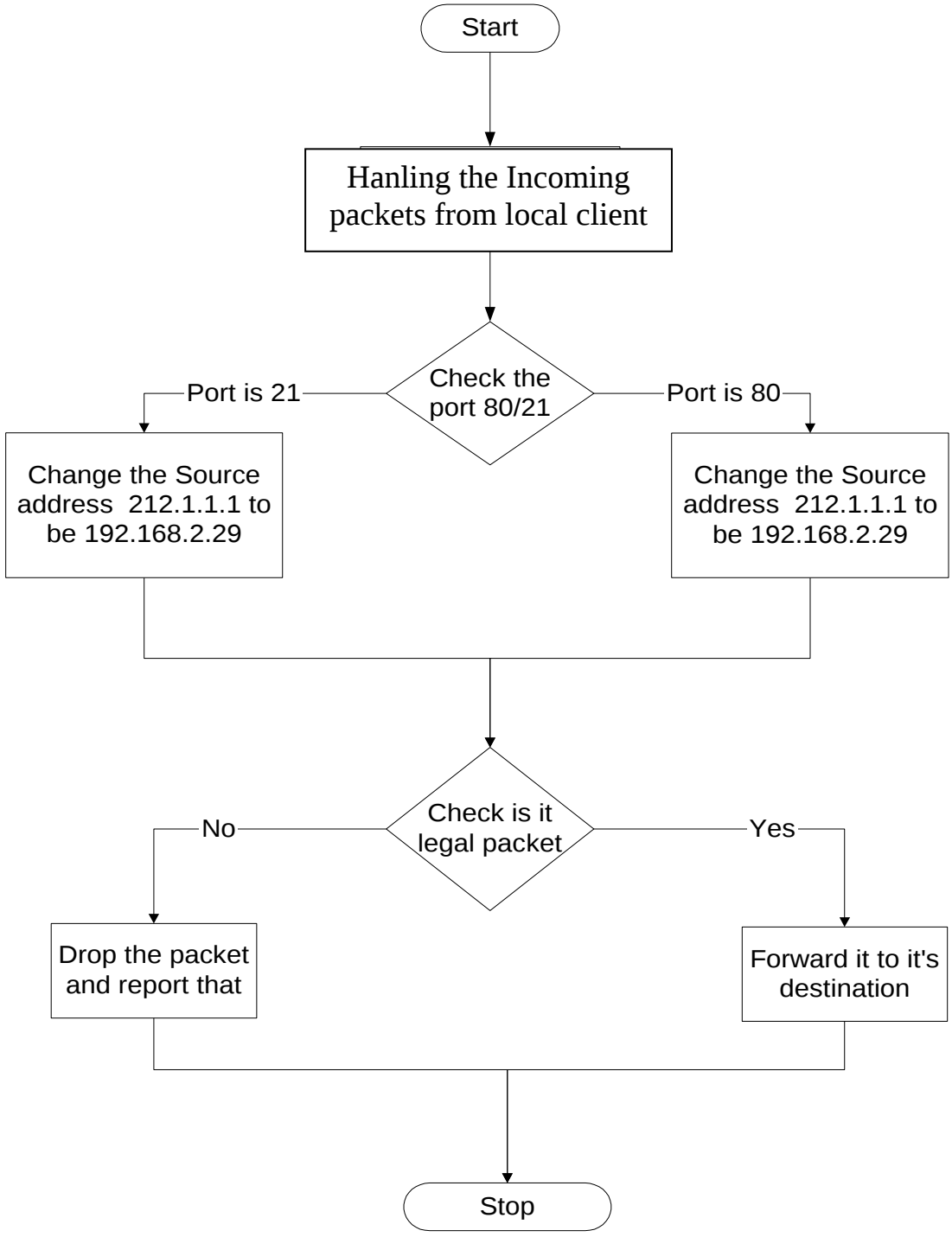


Figure 4.5: The Packet coming from local client to our FTP/HTTP Server

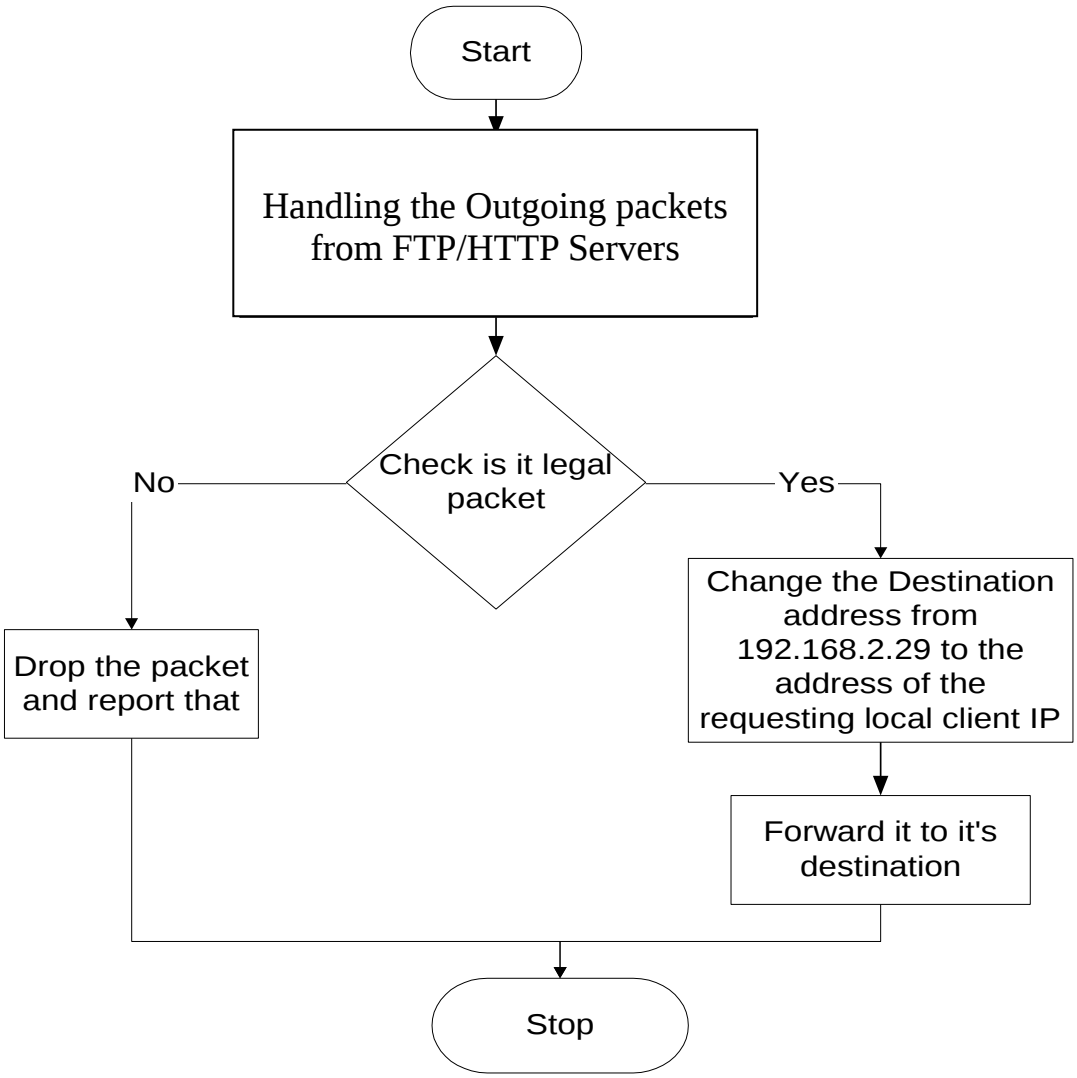


Figure 4.6: Flowchart of the packet from our FTP/HTTP server to Local client

5.5.4 Internal computer want connect to external server:

Now, we will consider what happens if the packet was instead generated by a client on the local network which wants to connect an external HTTP or FTP or any other server and it has the IP address **LAN_BOX**.

- Packet leaves **LAN_BOX** to **LAN_IP** because it is the default gateway.
- The packet reaches the firewall.
- Firewall check is it an illegal packet or not at the outgoing table?
- If it's not legal it will drop it or rejected according to the rule.

- If it is legal firewall change the **LAN_BOX** to **INET_IP** of the packet and send out to the Internet and keep information about the original source to redirect to it again.
- The packet leaves the firewall and reaches the external HTTP or FTP server. Packet reaches the HTTP or FTP server, and the HTTP or FTP box replies back to the firewall.
- Firewall check is it an illegal packet or not at the incoming table?
- If it's not legal it will drop it or rejected according to the rule.
- If it is legal firewall redirect the packet to the **LAN_BOX**.

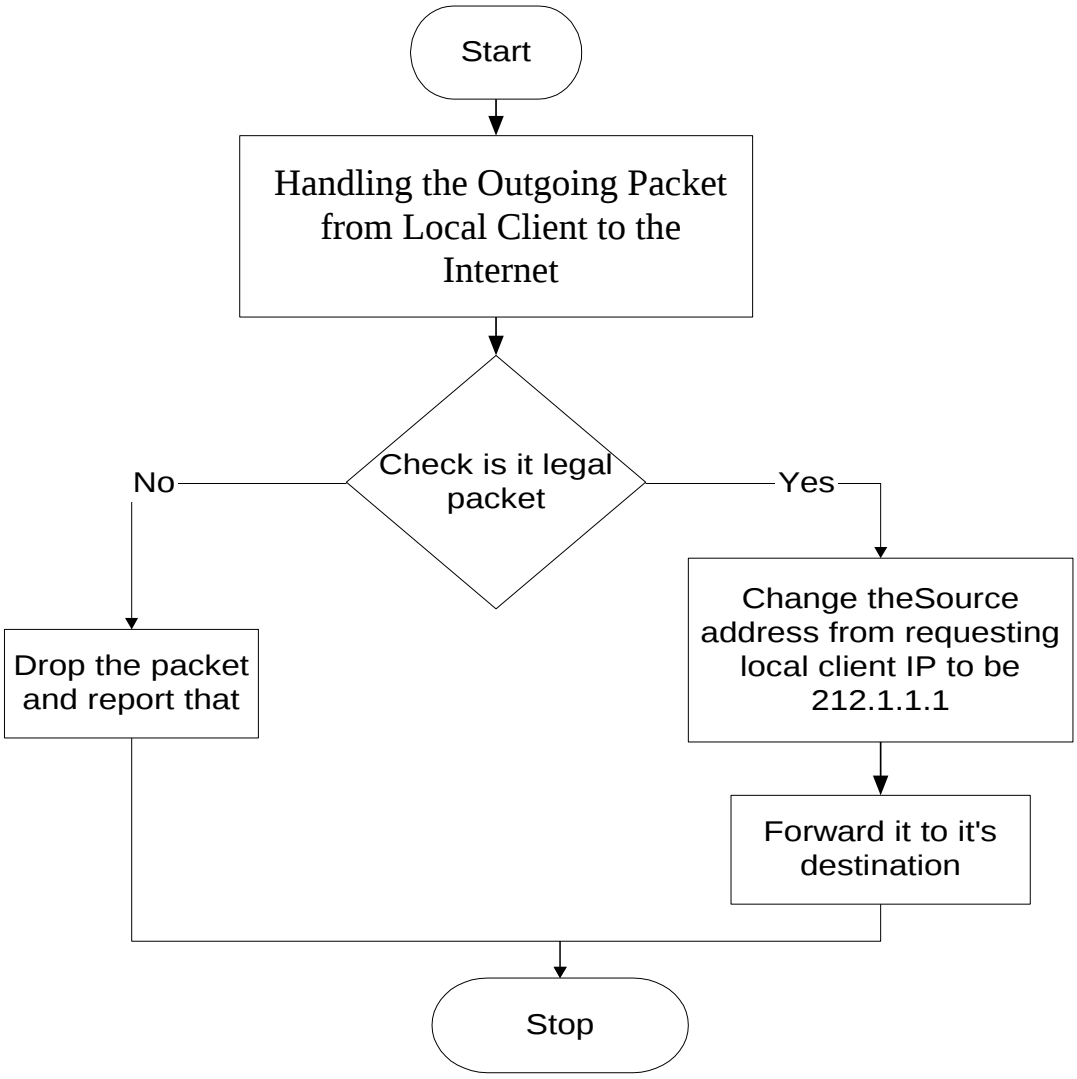


Figure 4.7: Flowchart for the packet from local client to the Internet

Chapter Five

6. CHAPTER FIVE: TESTING AND RESULTS

6.1 Test and Results

As shown in figures 5.1, and 5.2 below the testing network consist of two local area networks (LAN1 and LAN2) LAN1 consist of switch1, client1 – with IP address 192.168.2.2 - , and server1 – with IP address 192.168.2.28 – as FTP server, and the firewall (dual host) with net card one which has the IP address 192.168.2.29 and the other IP address is 212.1.1.1 and act as default gateway for LAN1.

LAN2 consist of switch1, client1 – with IP address 212.1.1.32 -, and server1 – with IP address 212.1.1.28 – as HTTP server. The protected network is LAN1 the outgoing and incoming packet goes through the firewall with its two NICs. The policies applied for incoming packets are:

num	target	protocol	source	destination	port
1	REJECT	tcp	212.1.1.32	192.168.2.28	tcp dpt:21
2	REJECT	tcp	64.1.39.2.30	192.168.2.28	tcp dpt:80
3	ACCEPT	udp	212.0.139.1	192.168.2.30	tcp dpt:21
4	DROP	icmp	212.0.139.1	192.168.2.28	tcp dpt:80
⋮	⋮	⋮	⋮	⋮	⋮

The policies applied for outgoing packet are:

num	target	protocol	source	destination
1	REJECT	tcp	192.168.2.2	212.1.1.28
2	REJECT	tcp	192.168.2.12	64.1.39.2.30
3	ACCEPT	udp	192.168.2.20	212.0.139.1
4	DROP	icmp	192.168.2.4	212.0.139.1
5	REJECT	all	192.168.2.5	212.0.139.1
⋮	⋮	⋮	⋮	⋮

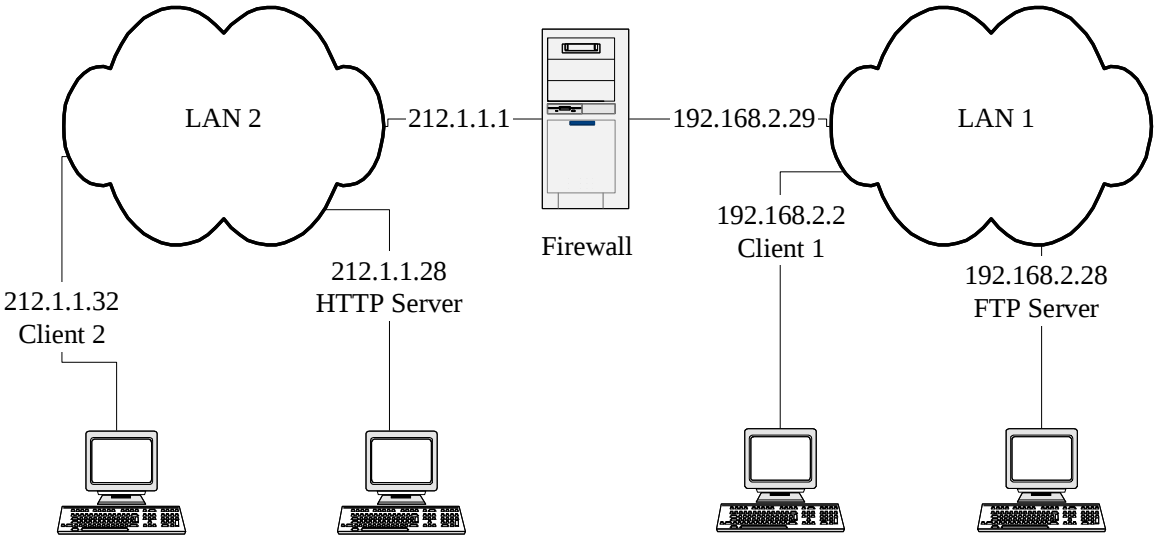


Figure 5.1: The Logical Testing Network

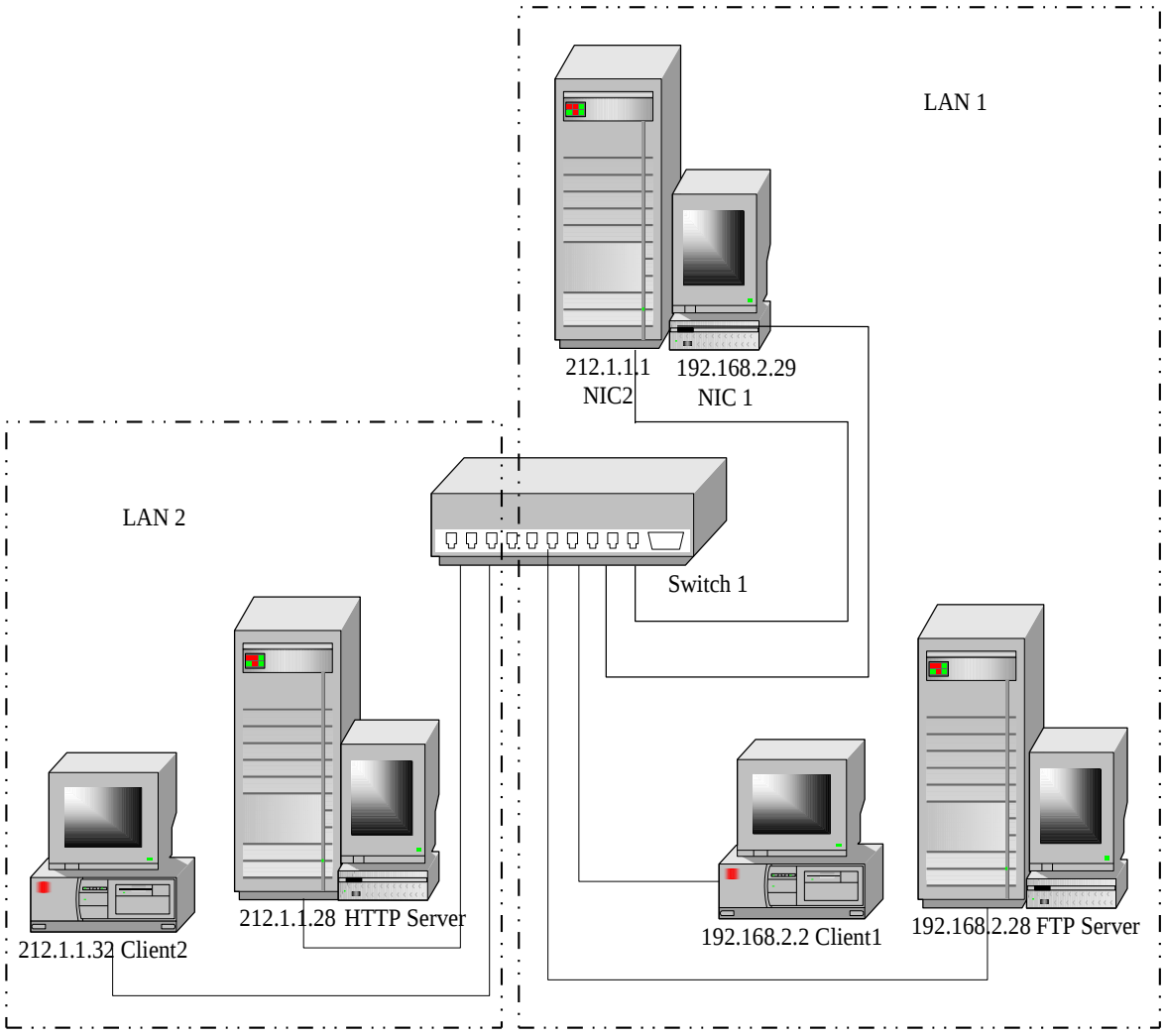
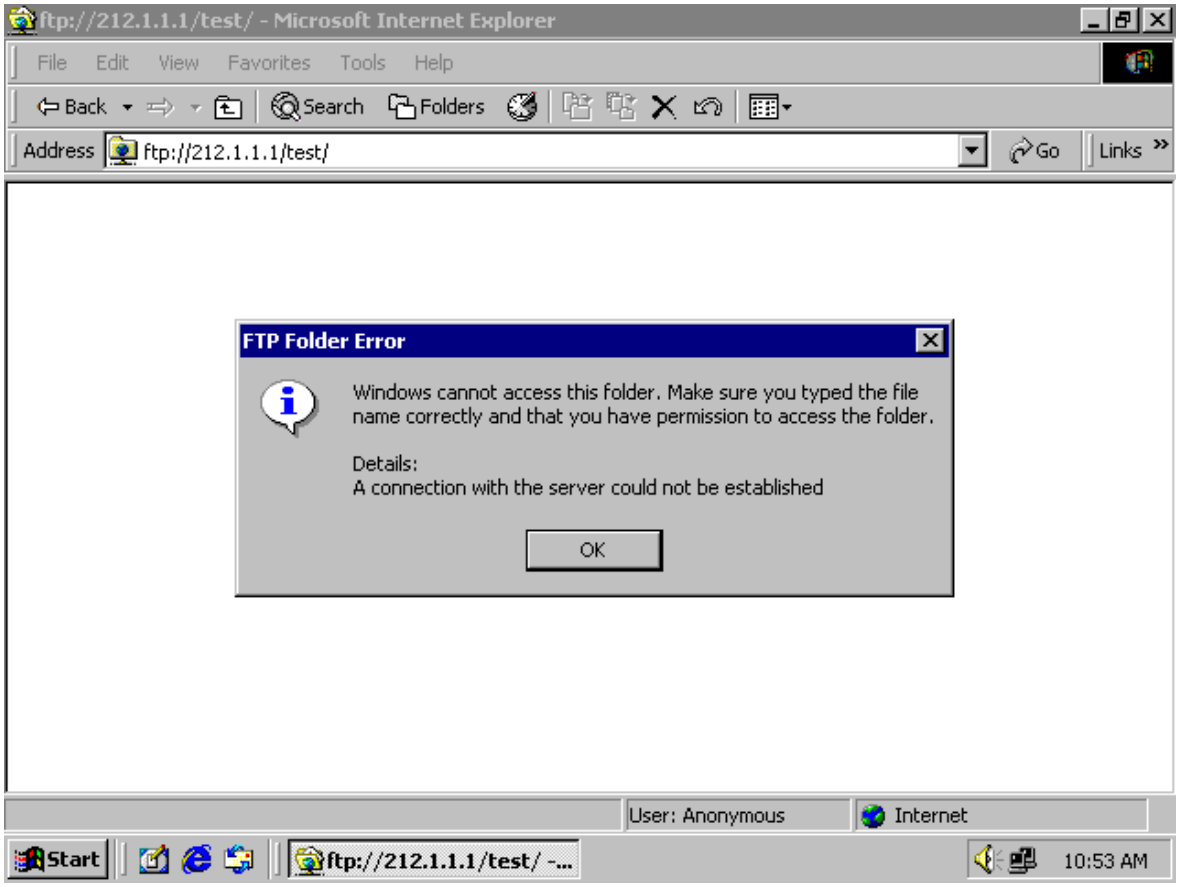


Figure 5.2: The Physical Testing Network

The screens that stated the work of the firewall have been captured as the testing results as shown below:

- The status of the client 212.1.1.32 before starting the firewall (can't establish a connection to our FTP server).
- The status of the client 212.1.1.32 after starting the firewall (can establish a connection to our FTP server and list its content).
- The status of the client 212.1.1.32 after applying the policy which rejects the tcp connection to our FTP server.
- The log File shows that the packets come from the client 212.1.1.32 to our FTP server was rejected.



.Figure 5.3: Before starting the Firewall

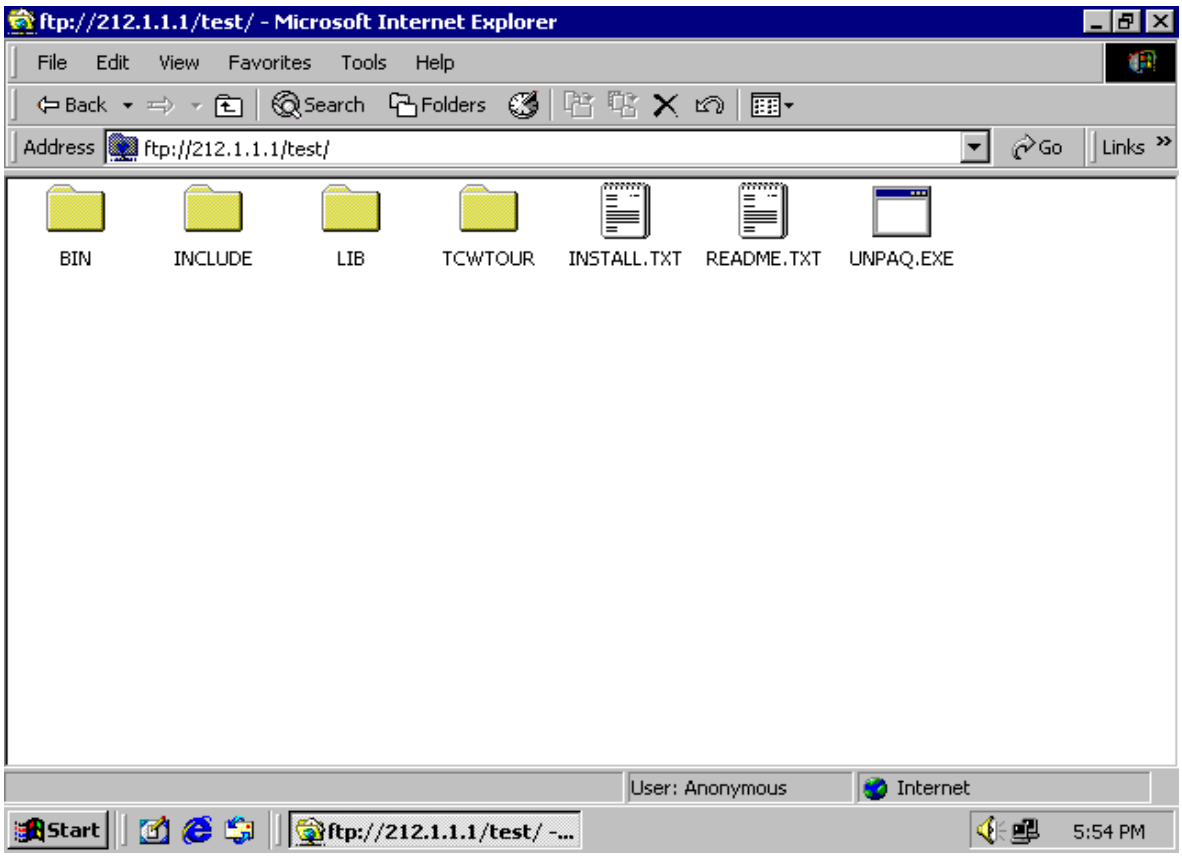
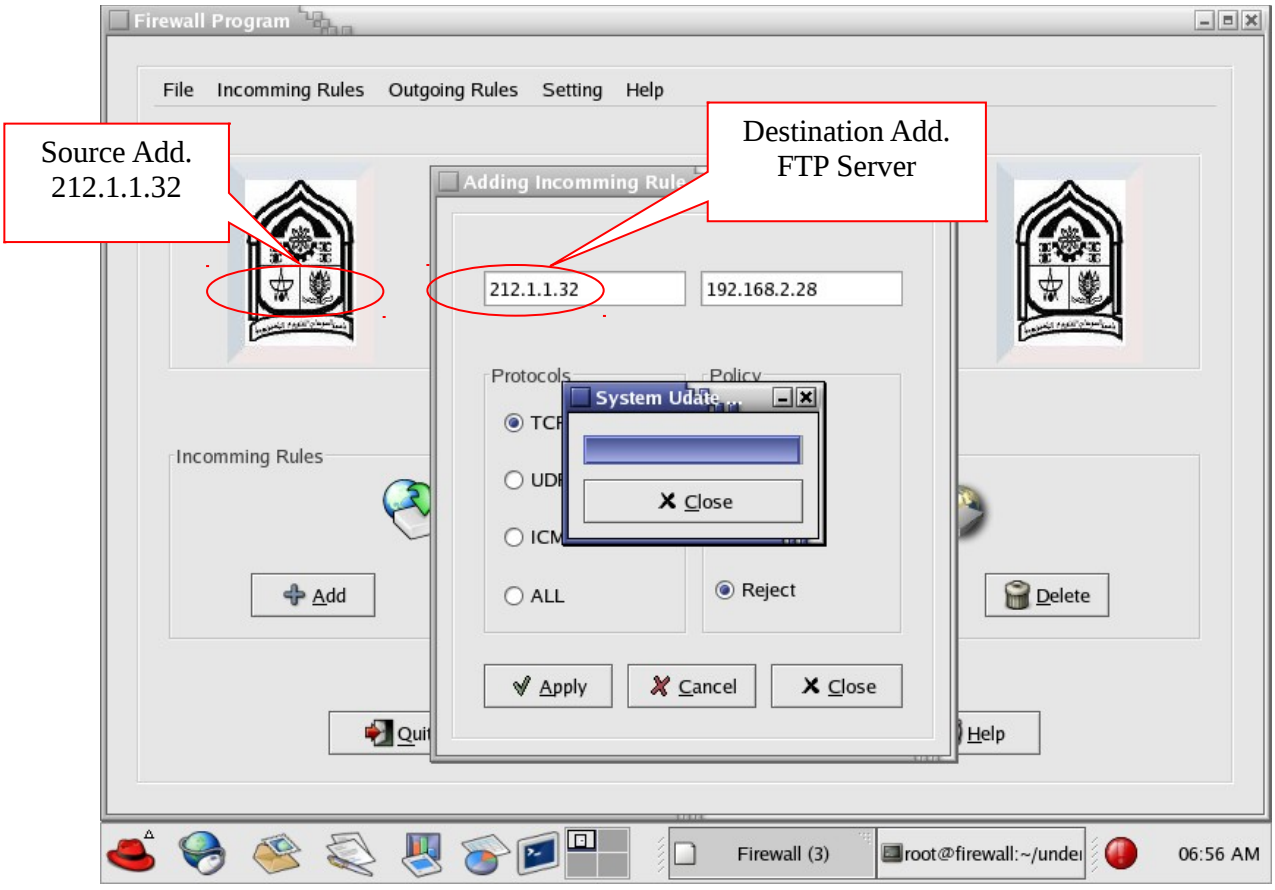
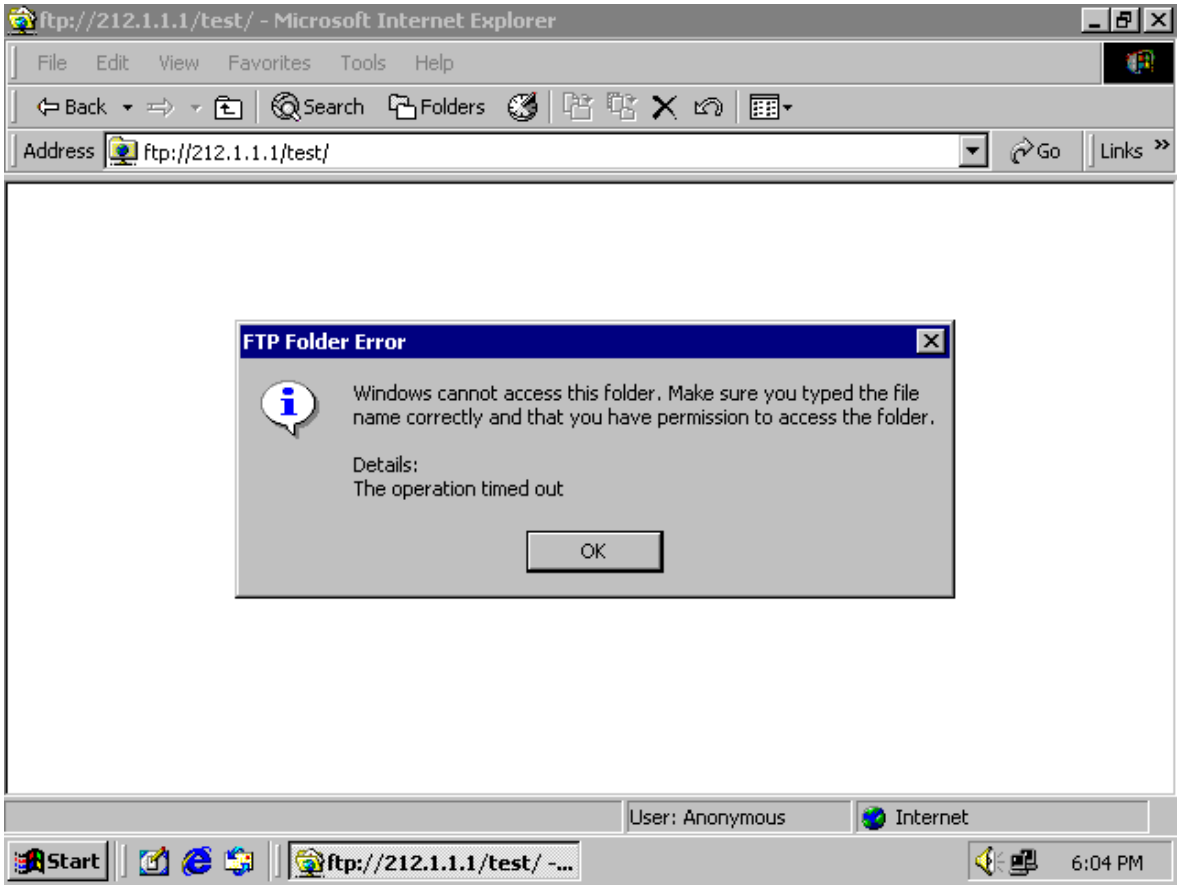


Figure 5.4: After starting the Firewall.



.Figure 5.5: Appling the Incoming policy



.Figure 5.6: After applying the policy

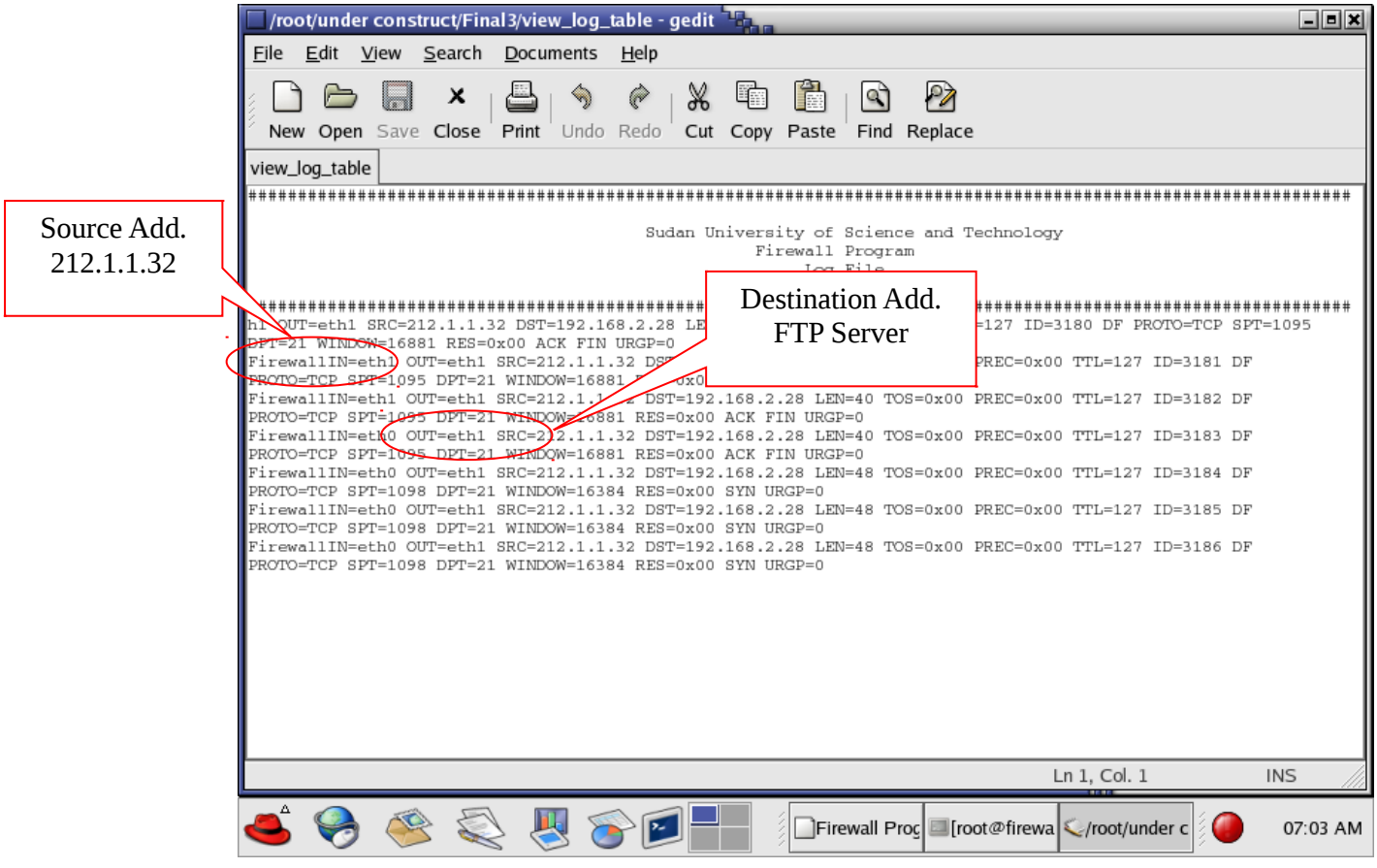


Figure 5.7: Log File after Appling the policy.

6.2 Conclusion

The study concluded that firewalls are necessary tools for data protection. Since firewall is an approach to security it requires a combination of component of hardware and software to control the flow of information within and between networks.

The goal of this research was to build and implement packet-filtering firewall. This was implemented by using the ordinary personal computer with Gimp Tool Kit (GTK), ordinary C–Language, and Red Hat Linux 8 as an Operating System. The software was implemented to deny some information as well as permitting others according to the policy that designed for that information.

Each firewall system serve specific function according to the security policy that you need it. Then, it can use more than one security policy with one-firewall techniques.

This research is elaborate on the packet filtering firewall technology that depends on the IP address and protocols for users in specific network. It considered as easier software and allows freedom for users to choose all available services in the network, that it is not assign in other techniques.

6.3 Recommendations

The Packet filtering was made as simple as possible. This software can be considered as a start point for further work related to network security programming for those who want to improve this application, this software dosen't support the Internet Protocol version 6 (IPv6); so the addressing scheme must be known well and the method for cheking the protocol version must be added to get software portability.

For those who want to develop this software the following points are considered as further work:

- This software filters TCP, UDP and ICMP packets and can be modified to filter the SMTP packets.
- Instead of working with IP addresses, the software can be modified to use servers name with the aid of DNS.
- This software is designed to work with Ethernet technology, but it can be modified to work with other technologies.

REFERENCES

1. D. Brent Chapman & Elizabeth D. Zwicky, Building Internet Firewall, O'Reilly & Associates, Inc., Second Edition, June 2000.
2. Christopher Negus, Red Hat Linux 8.0 Bible, Wiley Publishing, Inc., First Edition, 2003.
3. Mathew Strebe and Charles Perkins, Firewalls 24seven, SYBEX Inc., First Edition, 2000.
4. Bill Ball & Stephen Smoogen, *SAMS'* Teach Yourself LINUX in 24 Hours, *SAMS* Publishing and Red Hat Press, First Edition, 1998.
5. Kurt Wall, Mark Watson, and Mark Whitis, Linux Programming Unleashed, *SAMS* Publishing, First Edition, 1999.
6. Michael Timann, Official Red Hat Linux8 Administrator's Guide, Wiley Publishing, Inc, First Edition, 2003.
7. Mark Grennan, Firewall and Proxy Server HOWTO, http://www.grennan.com/, February 2000.
8. Kevin Fenzi & Dave Wreski, Linux Security-How-To, http://www.linuxdoc.org, February 2002.
9. Warren W. Gay, Linux Socket Programming By Examples, Macmillan Computer Publishing, Inc.,

APPENDIXES

Appendix A: Source Code

```
include<stdio.h>
include<netinet/in.h>
include<sys/types.h>
include<string.h>
include<arpa/inet.h>
include<netdb.h>
include<sys/socket.h>
include <gtk/gtk.h>
include<string.h>
struct ipaddresses
                    {
                     char HTTP[20];
                     char FTP[20];
                     char FLOCAL[20];
                     char FPUBLIC[20];
                     char SUB[30];
                    }x;
struct del
                   {
                   int no;
                   char in[120];
                   char flog[120];
                   char fin[120];
                   };
FILE *fp;
typedef struct _ProgressData {
GtkWidget *window;
GtkWidget *pbar;
int timer;
gboolean activity_mode;
} ProgressData;
GtkWidget *buttonp;
GtkWidget *vboxp;
char prot[8]=" -p all ";
char soip[24];
char policy[11]=" -j REJECT ";
char dsip[24];
void comand();
void comand1();
void act_tcp();
void act_udp();
void act_icmp();
void act_all();
```

```
void act_accept();
void act_drop();
void act_reject();
void get_soip( GtkWidget *widget,GtkWidget *entry );
void get_dsip( GtkWidget *widget,GtkWidget *entry );
void add_in_rule();
void add_out_rule();
void remove_in_rule();
void remove_out_rule();
void del_in_rule( GtkWidget *widget,GtkWidget *entry);
void del_out_rule( GtkWidget *widget,GtkWidget *entry);
void helpabout( );
static void menuitem_response( );
static void view_all_in();
static void view_all_out();
void save();
void reset();
void restore();
void reset_log();
void destroy_progress( GtkWidget *widget,ProgressData *pdata);
gint progress_timeout( gpointer data );
void error_sip();
void prog();
void view_backup();
void reset_backup();
void getFPUBLIC( GtkWidget *widget,GtkWidget *entry );
void getFLOCAL( GtkWidget *widget,GtkWidget *entry );
void getHTTP( GtkWidget *widget,GtkWidget *entry );
void getFTP( GtkWidget *widget,GtkWidget *entry );
void getSUB( GtkWidget *widget,GtkWidget *entry );
void apply();
void error_fpip();
void error_flip();
void error_ftp();
void error_http();
void entry_toggle_editable( GtkWidget *checkbutton,GtkWidget *entry );
void setting();
void help();
int main( int   argc,char *argv[] )
{
GtkWidget *window;
GtkWidget *frame;
GtkWidget *button;
GtkWidget *main_vbox;
GtkWidget *hbox;
GtkWidget *vbox;
GtkWidget *bbox;
GtkWidget *frame_vert;
GtkWidget *hbox1;
```

```
GtkWidget *vbox1;
GtkWidget *hbox2;
GtkWidget *frame1;
GtkWidget *label;
GtkWidget *menu;
GtkWidget *menu_bar;
GtkWidget *root_menu;
GtkWidget *root_menu1;
GtkWidget *menu1;
GtkWidget *root_menu2;
GtkWidget *menu2;
GtkWidget *root_menu3;
GtkWidget *menu3;
GtkWidget *root_menu4;
GtkWidget *menu4;
GtkWidget *menu_items;
GtkWidget *image;
gtk_init (&argc, &argv);
restore();
system(" insmod ip_conntrack_ftp ");
system("insmod ip_nat_ftp");
window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
gtk_window_set_title (GTK_WINDOW (window), "Firewall Program");
g_signal_connect (G_OBJECT (window), "destroy",
                                        G_CALLBACK (gtk_main_quit), NULL);
gtk_widget_set_size_request (window, 780, 530);
gtk_container_set_border_width (GTK_CONTAINER (window),0);
frame = gtk_frame_new (NULL);
gtk_container_add (GTK_CONTAINER (window), frame);
gtk_container_set_border_width (GTK_CONTAINER (frame), 20);
main_vbox = gtk_vbox_new (FALSE,10);
gtk_container_add (GTK_CONTAINER (frame), main_vbox);
gtk_container_set_border_width (GTK_CONTAINER (main_vbox), 0);
menu = gtk_menu_new ();
/*Create File Menu*/
menu_items = gtk_menu_item_new_with_label ("View Log File");
gtk_menu_shell_append (GTK_MENU_SHELL (menu), menu_items);
g_signal_connect_swapped (G_OBJECT (menu_items), "activate",
                          G_CALLBACK (menuitem_response),g_strdup (NULL));
gtk_widget_show (menu_items);
menu_items = gtk_menu_item_new_with_label ("Reset All Rules");
gtk_menu_shell_append (GTK_MENU_SHELL (menu), menu_items);
g_signal_connect_swapped (G_OBJECT (menu_items), "activate",
                                        G_CALLBACK (reset),g_strdup (NULL));
gtk_widget_show (menu_items);
menu_items = gtk_menu_item_new_with_label ("Reset Log File");
gtk_menu_shell_append (GTK_MENU_SHELL (menu), menu_items);
g_signal_connect_swapped (G_OBJECT (menu_items), "activate",
                                    G_CALLBACK (reset_log),g_strdup (NULL));
```

```
gtk_widget_show (menu_items);
menu_items = gtk_menu_item_new_with_label ("Exit");
gtk_menu_shell_append (GTK_MENU_SHELL (menu), menu_items);
gtk_widget_show (menu_items);
g_signal_connect_swapped (G_OBJECT (menu_items), "activate",
                                        G_CALLBACK (gtk_widget_destroy), window);
root_menu = gtk_menu_item_new_with_label ("File");
gtk_widget_show (root_menu);
gtk_menu_item_set_submenu (GTK_MENU_ITEM (root_menu), menu);
vbox = gtk_vbox_new (TRUE, 10);
gtk_container_add (GTK_CONTAINER (main_vbox), vbox);
gtk_widget_show (vbox);
menu_bar = gtk_menu_bar_new ();
gtk_box_pack_start (GTK_BOX (vbox), menu_bar, FALSE, FALSE, 4);
gtk_widget_show (menu_bar);
gtk_menu_shell_append (GTK_MENU_SHELL (menu_bar), root_menu);
menu1 = gtk_menu_new ();
menu_items = gtk_menu_item_new_with_label ("Add Rule");
gtk_menu_shell_append (GTK_MENU_SHELL (menu1), menu_items);
g_signal_connect_swapped (G_OBJECT (menu_items), "activate",
                                        G_CALLBACK (add_in_rule),g_strdup (NULL));
gtk_widget_show (menu_items);
menu_items = gtk_menu_item_new_with_label ("Remove Rule");
g_signal_connect_swapped (G_OBJECT (menu_items), "activate",
                                      G_CALLBACK (remove_in_rule),g_strdup (NULL));
gtk_menu_shell_append (GTK_MENU_SHELL (menu1), menu_items);
gtk_widget_show (menu_items);
menu_items = gtk_menu_item_new_with_label ("View All Rules");
gtk_menu_shell_append (GTK_MENU_SHELL (menu1), menu_items);
g_signal_connect_swapped (G_OBJECT (menu_items), "activate",
                                        G_CALLBACK (view_all_in),g_strdup (NULL));
gtk_widget_show (menu_items);
root_menu1 = gtk_menu_item_new_with_label ("Incomming Rules");
gtk_menu_item_set_submenu (GTK_MENU_ITEM (root_menu1), menu1);
gtk_box_pack_start (GTK_BOX (vbox), menu_bar, FALSE, FALSE, 4);
gtk_widget_show (menu_bar);
gtk_menu_shell_append (GTK_MENU_SHELL (menu_bar), root_menu1);
menu2 = gtk_menu_new ();
menu_items = gtk_menu_item_new_with_label ("Add Rule");
gtk_menu_shell_append (GTK_MENU_SHELL (menu2), menu_items);
g_signal_connect_swapped (G_OBJECT (menu_items), "activate",
                                      G_CALLBACK (add_out_rule),g_strdup (NULL));
gtk_widget_show (menu_items);
menu_items = gtk_menu_item_new_with_label ("Remove Rule");
gtk_menu_shell_append (GTK_MENU_SHELL (menu2), menu_items);
g_signal_connect_swapped (G_OBJECT (menu_items), "activate",
                                    G_CALLBACK (remove_out_rule),g_strdup (NULL));
gtk_widget_show (menu_items);
menu_items = gtk_menu_item_new_with_label ("View All Rules");
```

```
gtk_menu_shell_append (GTK_MENU_SHELL (menu2), menu_items);
g_signal_connect_swapped (G_OBJECT (menu_items), "activate",
                                        G_CALLBACK (view_all_out),g_strdup (NULL));
gtk_widget_show (menu_items);
root_menu2 = gtk_menu_item_new_with_label ("Outgoing Rules");
gtk_menu_item_set_submenu (GTK_MENU_ITEM (root_menu2), menu2);
gtk_box_pack_start (GTK_BOX (vbox), menu_bar, FALSE, FALSE, 0);
gtk_widget_show (menu_bar);
gtk_menu_shell_append (GTK_MENU_SHELL (menu_bar), root_menu2);
/*Create Setting  Menu*/
menu4 = gtk_menu_new ();
menu_items = gtk_menu_item_new_with_label ("Firewall Setting");
gtk_menu_shell_append (GTK_MENU_SHELL (menu4), menu_items);
g_signal_connect_swapped (G_OBJECT (menu_items), "activate",
                                          G_CALLBACK (setting),g_strdup (NULL));
gtk_widget_show (menu_items);
menu_items = gtk_menu_item_new_with_label ("View Log Backup File");
gtk_menu_shell_append (GTK_MENU_SHELL (menu4), menu_items);
g_signal_connect_swapped (G_OBJECT (menu_items), "activate",
                                        G_CALLBACK (view_backup),g_strdup (NULL));
gtk_widget_show (menu_items);
menu_items = gtk_menu_item_new_with_label ("Reset Log Backup File");
gtk_menu_shell_append (GTK_MENU_SHELL (menu4), menu_items);
g_signal_connect_swapped (G_OBJECT (menu_items), "activate",
                                        G_CALLBACK (reset_backup),g_strdup (NULL));
gtk_widget_show (menu_items);
root_menu4 = gtk_menu_item_new_with_label ("Setting");
gtk_menu_item_set_submenu (GTK_MENU_ITEM (root_menu4), menu4);
gtk_box_pack_start (GTK_BOX (vbox), menu_bar, FALSE, FALSE, 0);
gtk_widget_show (menu_bar);
gtk_menu_shell_append (GTK_MENU_SHELL (menu_bar), root_menu4);
/*Create Help Menu*/
menu3 = gtk_menu_new ();
menu_items = gtk_menu_item_new_with_label ("How To Use");
gtk_menu_shell_append (GTK_MENU_SHELL (menu3), menu_items);
g_signal_connect_swapped (G_OBJECT (menu_items), "activate",
                                            G_CALLBACK (help),g_strdup (NULL));
gtk_widget_show (menu_items);
menu_items = gtk_menu_item_new_with_label ("About Firewall");
gtk_menu_shell_append (GTK_MENU_SHELL (menu3), menu_items);
g_signal_connect_swapped (G_OBJECT (menu_items), "activate",
                                          G_CALLBACK (helpabout),g_strdup (NULL));
gtk_widget_show (menu_items);
root_menu3 = gtk_menu_item_new_with_label ("Help");
gtk_menu_item_set_submenu (GTK_MENU_ITEM (root_menu3), menu3);
gtk_box_pack_start (GTK_BOX (vbox), menu_bar, FALSE, FALSE, 0);
gtk_widget_show (menu_bar);
gtk_menu_shell_append (GTK_MENU_SHELL (menu_bar), root_menu3);
/*Create Frame for the Program lable*/40
```

```
frame = gtk_frame_new (NULL);
hbox = gtk_hbox_new (FALSE,10);
gtk_container_add (GTK_CONTAINER (frame), hbox);
image = gtk_image_new_from_file ("sust.jpg");
gtk_container_add (GTK_CONTAINER (hbox), image);
gtk_box_pack_start(GTK_BOX(frame),hbox,FALSE,FALSE,0);
gtk_widget_show(image);
label = gtk_label_new ("\n\nSudan University of Science and Technolog\n\tCollege
of Post Graduate Studies\n\t\tThis is Firewall Program\n\t    Created by:  Isam
Abdelnabi\n\tSupervised by: Dr. Yahia Abdalla\n\n");
gtk_container_add (GTK_CONTAINER (hbox), label);
gtk_container_set_border_width (GTK_CONTAINER (frame), 20);
gtk_box_pack_start (GTK_BOX (main_vbox), frame, FALSE, FALSE, 0);
image = gtk_image_new_from_file ("sust.jpg");
gtk_container_add (GTK_CONTAINER (hbox), image);
gtk_box_pack_start(GTK_BOX(frame),hbox,FALSE,FALSE,0);
gtk_widget_show(image);
/*************************************************************/
            /*Creat Button box for Incomming Rules*/
/*************************************************************/
hbox1 = gtk_hbox_new(TRUE,0);
gtk_box_pack_start (GTK_BOX (main_vbox), hbox1, TRUE, TRUE, 0);
frame1 = gtk_frame_new("Incomming Rules");
gtk_container_add (GTK_CONTAINER (hbox1), frame1);
gtk_container_set_border_width (GTK_CONTAINER (frame1), 20);
vbox1 = gtk_vbox_new (TRUE, 0);
gtk_container_add (GTK_CONTAINER (frame1), vbox1);
gtk_container_set_border_width (GTK_CONTAINER (vbox), 10);
hbox2 = gtk_hbox_new(TRUE,0);
image = gtk_image_new_from_file ("test.png");
gtk_container_add (GTK_CONTAINER (hbox2), image);
gtk_box_pack_start(GTK_BOX(vbox1),hbox2,FALSE,FALSE,0);
gtk_widget_show(image);
bbox = gtk_hbutton_box_new ();
gtk_container_set_border_width (GTK_CONTAINER (bbox), 5);
gtk_container_add (GTK_CONTAINER (vbox1), bbox);
/* Set the appearance of the Button Box */
gtk_button_box_set_layout (GTK_BUTTON_BOX
(bbox),GTK_BUTTONBOX_SPREAD);
gtk_box_set_spacing (GTK_BOX (bbox), 10);
/*gtk_button_box_set_child_size (GTK_BUTTON_BOX (bbox), child_w,
child_h);*/
button = gtk_button_new_from_stock (GTK_STOCK_ADD);
gtk_container_add (GTK_CONTAINER (bbox), button);
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                                    G_CALLBACK (add_in_rule),g_strdup (NULL) );
button = gtk_button_new_from_stock (GTK_STOCK_DELETE);
gtk_container_add (GTK_CONTAINER (bbox), button);
```

```
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                                    G_CALLBACK (remove_in_rule),g_strdup (NULL) );
gtk_box_pack_start (GTK_BOX (hbox),frame1,TRUE,TRUE,10);
frame1 = gtk_frame_new("Outgoing Rules");
gtk_container_add (GTK_CONTAINER (hbox1), frame1);
gtk_container_set_border_width (GTK_CONTAINER (frame1), 20);
vbox1 = gtk_vbox_new (TRUE, 0);
gtk_container_add (GTK_CONTAINER (frame1), vbox1);
gtk_container_set_border_width (GTK_CONTAINER (vbox), 10);
hbox2 = gtk_hbox_new(TRUE,0);
image = gtk_image_new_from_file ("test1.png");
gtk_container_add (GTK_CONTAINER (hbox2), image);
gtk_box_pack_start(GTK_BOX(vbox1),hbox2,FALSE,FALSE,0);
gtk_widget_show(image);
bbox = gtk_hbutton_box_new ();
gtk_container_set_border_width (GTK_CONTAINER (bbox), 20);
gtk_container_add (GTK_CONTAINER (vbox1), bbox);
/* Set the appearance of the Button Box */
gtk_button_box_set_layout (GTK_BUTTON_BOX
(bbox),GTK_BUTTONBOX_SPREAD);
gtk_box_set_spacing (GTK_BOX (bbox), 10);
/*gtk_button_box_set_child_size (GTK_BUTTON_BOX (bbox), child_w,
child_h);*/
button = gtk_button_new_from_stock (GTK_STOCK_ADD);
gtk_container_add (GTK_CONTAINER (bbox), button);
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                                     G_CALLBACK (add_out_rule),g_strdup (NULL) );
button = gtk_button_new_from_stock (GTK_STOCK_DELETE);
gtk_container_add (GTK_CONTAINER (bbox), button);
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                                    G_CALLBACK (remove_out_rule),g_strdup (NULL) );
gtk_box_pack_start (GTK_BOX (hbox),frame1,TRUE,TRUE,10);
/*************************************************************/
          /*Create Button Box for the Whole Application*/
/*************************************************************/
frame_vert = gtk_hbox_new(FALSE,0);
gtk_box_pack_start (GTK_BOX (main_vbox), frame_vert, TRUE, TRUE, 0);
hbox = gtk_hbox_new (FALSE, 0);
gtk_container_set_border_width (GTK_CONTAINER (hbox), 10);
gtk_container_add (GTK_CONTAINER (frame_vert), hbox);
bbox = gtk_hbutton_box_new ();
gtk_container_set_border_width (GTK_CONTAINER (bbox), 20);
gtk_container_add (GTK_CONTAINER (hbox), bbox);
/* Set the appearance of the Button Box */
gtk_button_box_set_layout (GTK_BUTTON_BOX
                                    (bbox),GTK_BUTTONBOX_SPREAD);
gtk_box_set_spacing (GTK_BOX (bbox),10);
button = gtk_button_new_from_stock (GTK_STOCK_QUIT);
gtk_container_add (GTK_CONTAINER (bbox), button);
```

```
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                                                    G_CALLBACK (save), NULL);
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                                          G_CALLBACK (gtk_widget_destroy), window);
/*Create a View Log File Button for a Whole Application*/
button = gtk_button_new_with_label ("View Log File");
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                                G_CALLBACK (menuitem_response),g_strdup (NULL) );
gtk_container_add (GTK_CONTAINER (bbox), button);
/*Creat A Help Button*/
button = gtk_button_new_from_stock (GTK_STOCK_HELP);
gtk_container_add (GTK_CONTAINER (bbox), button);
gtk_box_pack_start (GTK_BOX (hbox),frame,TRUE,TRUE,10);
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                                              G_CALLBACK (help),g_strdup (NULL) );
/* Display the window */
gtk_widget_show_all (window);
/* Enter the event loop */
gtk_main ();
system("clear");
return 0;
}
void comand()
{
char temp1[120]="",temp2[120]="";
char in[120]= "iptables -A INPUT ";
char inf[120]= "iptables -A FORWARD  ";
char logf[120]= "iptables -A FORWARD  ";
char din[120]= "iptables -D INPUT ";
char dinf[120]= "iptables -D FORWARD  ";
char dlogf[120]= "iptables -D FORWARD  ";
struct hostent *ptrh;
struct hostent *ptrh1;
struct del delin;
fp=fopen("setting.dat","r");
while(!feof(fp))
            {
                 fread(&x,sizeof x,1,fp);
            }
fclose(fp);
ptrh = gethostbyname(dsip);
ptrh1 = gethostbyname(soip);
if((((char*)ptrh) == NULL)||((char*)ptrh1) == NULL)
        printf("error");
else
            {
             strcat(temp1," -s ");
             strcat(temp1,soip);
             strcat(temp1," -d ");
```

```
            strcat(temp1,dsip);
            if(strcmp(dsip,x.HTTP)==0||strcmp(dsip,x.FTP)==0)
                  strncat(temp1," -p tcp ",8);
            else
            strncat(temp1,prot,8);
            strcpy(temp2,temp1);
            strncat(temp1,policy,10);
            strcat(in,temp1);
            strcat(din,temp1);
            strcat(temp2," -j LOG --log-prefix Firewall --log-ip-options");
            strcat(inf,temp1);
            strcat(dinf,temp1);
            strcat(logf,temp2);
            strcat(dlogf,temp2);
            if(strcmp(dsip,x.HTTP)==0)
             {
               strcat(in," --dport 80 ");
               strcat(din," --dport 80 ");
               strcat(logf," --dport 80 ");
               strcat(dlogf," --dport 80 ");
               strcat(inf," --dport 80 ");
               strcat(dinf," --dport 80 ");
             }
            if(strcmp(dsip,x.FTP)==0)
             {
               strcat(in," --dport 21 ");
               strcat(din," --dport 21 ");
               strcat(logf," --dport 21 ");
               strcat(dlogf," --dport 21 ");
               strcat(inf," --dport 21 ");
               strcat(dinf," --dport 21 ");
             }
          system(in);
          system(logf);
          system(inf);
          save();
          fp=fopen("delin.dat","a");
          delin.no = 1;
          strcpy(delin.in,din);
          strcpy(delin.flog,dlogf);
          strcpy(delin.fin,dinf);
          fwrite(&delin,sizeof delin,1,fp);
          fclose(fp);
          save();
          prog();
           }
}
void comand1()
{
```

```
char temp1[120]="",temp2[120]="";
char in[120]= "iptables -A OUTPUT ";
char inf[120]= "iptables -A FORWARD  ";
char logf[120]= "iptables -A FORWARD  ";
char din[120]= "iptables -D OUTPUT ";
char dinf[120]= "iptables -D FORWARD  ";
char dlogf[120]= "iptables -D FORWARD  ";
struct hostent *ptrh;
struct hostent *ptrh1;
struct del delin;
fp=fopen("setting.dat","r");
while(!feof(fp))
          {
                fread(&x,sizeof x,1,fp);
           }
fclose(fp);
ptrh = gethostbyname(dsip);
ptrh1 = gethostbyname(soip);
if((((char*)ptrh) == NULL)||((char*)ptrh1) == NULL)
          printf("error");
else
           {
            strcat(temp1," -s ");
            strcat(temp1,soip);
            strcat(temp1," -d ");
            strcat(temp1,dsip);
            if(strcmp(dsip,x.HTTP)==0||strcmp(dsip,x.FTP)==0)
                  strncat(temp1," -p tcp ",8);
            else
            strncat(temp1,prot,8);
            strcpy(temp2,temp1);
            strncat(temp1,policy,10);
            strcat(in,temp1);
            strcat(din,temp1);
            strcat(temp2," -j LOG --log-prefix Firewall --log-ip-options");
            strcat(inf,temp1);
            strcat(dinf,temp1);
            strcat(logf,temp2);
            strcat(dlogf,temp2);
            if(strcmp(dsip,x.HTTP)==0)
            {
            strcat(in," --dport 80 ");
            strcat(din," --dport 80 ");
            strcat(logf," --dport 80 ");
            strcat(dlogf," --dport 80 ");
            strcat(inf," --dport 80 ");
            strcat(dinf," --dport 80 ");
            }
            if(strcmp(dsip,x.FTP)==0)
```

```
                {
                strcat(in," --dport 21 ");
                strcat(din," --dport 21 ");
                strcat(logf," --dport 21 ");
                strcat(dlogf," --dport 21 ");
                strcat(inf," --dport 21 ");
                 strcat(dinf," --dport 21 ");
                 }
                 system(in);
                 system(logf);
                 system(inf);
                 save();
                fp=fopen("delout.dat","a");
                delin.no = 1;
                strcpy(delin.in,din);
                strcpy(delin.flog,dlogf);
                strcpy(delin.fin,dinf);
                fwrite(&delin,sizeof delin,1,fp);
                fclose(fp);
                save();
                prog();
                }
}
void reset_log()
{
system("dmesg -c -s 16392 | grep Firewall> Backup_of_log_file.txt");
prog();
}
void act_tcp()
{
        strcpy(prot," -p tcp ");
}
void act_udp()
{
        strcpy(prot," -p udp ");
}
void act_icmp()
{
        strcpy(prot," -p icmp ");
}
void act_all()
{
        strcpy(prot," -p all ");
}
void act_accept()
{
        strcpy(policy," -j ACCEPT ");
}
void act_drop()
```

```
{
            strcpy(policy,"  -j DROP  ");
}
void act_reject()
{
                strcpy(policy," -j REJECT ");
}

void get_soip(GtkWidget *widget,GtkWidget *entry)
{
struct hostent *ptrh;
const gchar *entry_text;
entry_text = gtk_entry_get_text (GTK_ENTRY (entry));
strcpy(soip,entry_text);
ptrh = gethostbyname(soip);
if(((char*)ptrh) == NULL)
         {
            error_sip();
          }
}
void get_dsip( GtkWidget *widget,GtkWidget *entry )
{
struct hostent *ptrh;
const gchar *entry_text;
entry_text = gtk_entry_get_text (GTK_ENTRY (entry));
 strcpy(dsip,entry_text);
 ptrh = gethostbyname(dsip);
 if(((char*)ptrh) == NULL)
       {
           error_dip();
       }
}
void error_sip()
{
 GtkWidget *dialog;
 GtkWidget *window1;
 char text[50]=("Error Invalid Source Address: ");
 strcat(text,soip);
 window1 = gtk_window_new (GTK_WINDOW_TOPLEVEL);
dialog=gtk_message_dialog_new(window1,
          GTK_DIALOG_DESTROY_WITH_PARENT,GTK_MESSAGE_ERROR,
GTK_BUTTONS_CLOSE,text);
gtk_window_set_position (dialog, GTK_WIN_POS_CENTER_ALWAYS);
gtk_dialog_run (GTK_DIALOG (dialog));
gtk_widget_destroy (dialog);
}
void error_dip()
{
GtkWidget *dialog1;
```

```
GtkWidget *window1;
char text[50]=("Error Invalid Destination Address: ");
strcat(text,dsip);
window1 = gtk_window_new (GTK_WINDOW_TOPLEVEL);
dialog1 = gtk_message_dialog_new (window1,
        GTK_DIALOG_DESTROY_WITH_PARENT,GTK_MESSAGE_ERROR,
GTK_BUTTONS_CLOSE,text);
gtk_window_set_position (dialog1, GTK_WIN_POS_CENTER_ALWAYS);
gtk_dialog_run (GTK_DIALOG (dialog1));
gtk_widget_destroy (dialog1);
}
void add_in_rule()
{
/* GtkWidget is the storage type for widgets */
GtkWidget *window;
GtkWidget *frame;
GtkWidget *entry;
GtkWidget *entry1;
GtkWidget *vbox;
GtkWidget *hbox;
GtkWidget *box2;
GtkWidget *box1;
GtkWidget *main_box;
GtkWidget *button;
GtkWidget *button1;
GtkWidget *button2;
GtkWidget *button3;
GtkWidget *button4;
GtkWidget *button5;
GtkWidget *button6;
GtkWidget *button7;
GtkWidget *label;
GSList *group;
GSList *group1;
gint tmp_pos;
window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
gtk_window_set_title (GTK_WINDOW (window), "Adding Incomming Rule");
gtk_widget_set_size_request (window, 350, 380);
gtk_window_set_position (window, GTK_WIN_POS_CENTER_ALWAYS);
gtk_window_set_resizable(window,FALSE);
/* Sets the border width of the window. */
gtk_container_set_border_width (GTK_CONTAINER (window), 10);
/* Create a Frame */
frame = gtk_frame_new (NULL);
gtk_container_add (GTK_CONTAINER (window), frame);
gtk_box_pack_start (GTK_BOX (window), frame, FALSE, FALSE, 10);
gtk_widget_show (frame);
/*Show Enery Boxes*/
main_box = gtk_vbox_new (FALSE, 0);
```

```
gtk_container_add (GTK_CONTAINER (frame), main_box);
gtk_container_set_border_width (GTK_CONTAINER (main_box), 10);
gtk_box_pack_start (GTK_BOX (frame), main_box, FALSE, FALSE, 10);
gtk_widget_show (main_box);
hbox = gtk_hbox_new (TRUE, 0);
label = gtk_label_new ("Source IP Address:        Distenation IP Address:");
gtk_container_add (GTK_CONTAINER (hbox), label);
gtk_container_set_border_width (GTK_CONTAINER (hbox), 10);
hbox = gtk_hbox_new (FALSE, 10);
gtk_container_add (GTK_CONTAINER (main_box), hbox);
gtk_container_set_border_width (GTK_CONTAINER (hbox), 10);
gtk_box_pack_start (GTK_BOX (main_box), hbox, FALSE, FALSE, 0);
gtk_widget_show (hbox);
vbox = gtk_vbox_new (FALSE, 0);
gtk_container_add (GTK_CONTAINER (hbox), vbox);
gtk_box_pack_start (GTK_BOX (hbox), vbox, FALSE, FALSE, 0);
gtk_widget_show (vbox);
entry = gtk_entry_new ();
gtk_entry_set_max_length (GTK_ENTRY (entry), 20);
gtk_entry_set_text (GTK_ENTRY (entry), "Sour. IP");
tmp_pos = GTK_ENTRY (entry)->text_length;
gtk_editable_insert_text (GTK_EDITABLE (entry), " Address", -1, &tmp_pos);
gtk_editable_select_region (GTK_EDITABLE (entry),0, GTK_ENTRY
(entry)->text_length);
gtk_box_pack_start (GTK_BOX (vbox), entry, TRUE, TRUE,0);
gtk_widget_show (entry);
vbox = gtk_vbox_new (FALSE, 0);
gtk_container_add (GTK_CONTAINER (hbox), vbox);
gtk_widget_show (vbox);
entry1 = gtk_entry_new ();
gtk_entry_set_max_length (GTK_ENTRY (entry1), 20);
gtk_entry_set_text (GTK_ENTRY (entry1), "Dist. IP ");
tmp_pos = GTK_ENTRY (entry1)->text_length;
gtk_editable_insert_text (GTK_EDITABLE (entry1), "Address", -1, &tmp_pos);
gtk_editable_select_region (GTK_EDITABLE (entry1),0, GTK_ENTRY
(entry1)->text_length);
gtk_box_pack_start (GTK_BOX (vbox), entry1, TRUE, TRUE,0);
gtk_widget_show (entry1);
 /*Create Protocols Radiobutton Box*/
box1 = gtk_hbox_new (FALSE, 10);
gtk_container_add (GTK_CONTAINER (main_box), box1);
gtk_container_set_border_width (GTK_CONTAINER (box1), 10);
gtk_box_pack_start (GTK_BOX (main_box), box1, FALSE, FALSE, 10);
gtk_widget_show (box1);
frame = gtk_frame_new ("Protocols");
gtk_container_add (GTK_CONTAINER (box1), frame);
gtk_box_pack_start (GTK_BOX (box1), frame, FALSE, FALSE, 10);
gtk_widget_show (frame);
box2 = gtk_vbox_new (FALSE, 10);
```

```
gtk_container_set_border_width (GTK_CONTAINER (box2), 10);
gtk_container_add (GTK_CONTAINER (frame), box2);
gtk_box_pack_start (GTK_BOX (frame), box2, TRUE, TRUE, 20);
gtk_widget_show (box2);
button = gtk_radio_button_new_with_label (NULL, "TCP");
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
g_signal_connect (G_OBJECT (button), "pressed",
                                                G_CALLBACK (act_tcp),NULL);
gtk_widget_show (button);
group = gtk_radio_button_get_group (GTK_RADIO_BUTTON (button));
button1 = gtk_radio_button_new_with_label (group, "UDP");
g_signal_connect (G_OBJECT (button1), "pressed",
                                                G_CALLBACK (act_udp),NULL);
gtk_box_pack_start (GTK_BOX (box2), button1, TRUE, TRUE, 0);
gtk_widget_show (button1);
button2 = gtk_radio_button_new_with_label_from_widget (
                                        GTK_RADIO_BUTTON (button),"ICMP");
gtk_box_pack_start (GTK_BOX (box2), button2, TRUE, TRUE, 0);
g_signal_connect (G_OBJECT (button2), "pressed",
                                                G_CALLBACK (act_icmp),NULL);
gtk_widget_show (button2);
button3 = gtk_radio_button_new_with_label_from_widget (
                                        GTK_RADIO_BUTTON (button),"ALL");
gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (button3), TRUE);
g_signal_connect (G_OBJECT (button3), "pressed",
                                                G_CALLBACK (act_all),NULL);
gtk_box_pack_start (GTK_BOX (box2), button3, TRUE, TRUE, 0);
gtk_widget_show (button3);
/*Create Polcy Button Box*/
frame = gtk_frame_new ("Policy");
gtk_container_add (GTK_CONTAINER (box1), frame);
gtk_box_pack_start (GTK_BOX (box1), frame, FALSE, FALSE, 0);
gtk_widget_show (frame);
box2 = gtk_vbox_new (FALSE, 10);
gtk_container_set_border_width (GTK_CONTAINER (box2), 5);
gtk_container_add (GTK_CONTAINER (frame), box2);
gtk_box_pack_start (GTK_BOX (frame), box2, TRUE, TRUE, 20);
gtk_widget_show (box2);
button4 = gtk_radio_button_new_with_label (NULL, "Accept");
gtk_box_pack_start (GTK_BOX (box2), button4, TRUE, TRUE, 0);
g_signal_connect (G_OBJECT (button4), "pressed",
                                                G_CALLBACK (act_accept),NULL);
gtk_widget_show (button4);
group1 = gtk_radio_button_get_group (GTK_RADIO_BUTTON (button4));
button5 = gtk_radio_button_new_with_label (group1, "Drop");
g_signal_connect (G_OBJECT (button5), "pressed",
                                                G_CALLBACK (act_drop),NULL);
gtk_box_pack_start (GTK_BOX (box2), button5, TRUE, TRUE, 0);
gtk_widget_show (button5);
```

```
button6 = gtk_radio_button_new_with_label_from_widget (
                                        GTK_RADIO_BUTTON (button4),"Reject");
gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (button6), TRUE);
g_signal_connect (G_OBJECT (button6), "pressed",
                                                G_CALLBACK (act_reject),NULL);
gtk_box_pack_start (GTK_BOX (box2), button6, TRUE, TRUE, 0);
gtk_widget_show (button6);
box2 = gtk_hbox_new (TRUE, 10);
gtk_container_set_border_width (GTK_CONTAINER (box2), 10);
gtk_box_pack_start (GTK_BOX (main_box), box2, FALSE, TRUE, 0);
gtk_widget_show (box2);
button7= gtk_button_new_from_stock (GTK_STOCK_APPLY);
gtk_box_pack_start (GTK_BOX (box2), button7, TRUE, TRUE, 0);
g_signal_connect (G_OBJECT (button7), "clicked",
                                                  G_CALLBACK(get_soip),entry);
g_signal_connect (G_OBJECT (button7), "clicked",
                                                G_CALLBACK (get_dsip),entry1);
g_signal_connect (G_OBJECT (button7), "clicked",
                                                 G_CALLBACK (comand),NULL);
gtk_widget_show (button);
button = gtk_button_new_from_stock (GTK_STOCK_CANCEL);
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
GTK_WIDGET_SET_FLAGS (button, GTK_CAN_DEFAULT);
gtk_widget_grab_default (button);
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                                        G_CALLBACK (gtk_widget_destroy), window);
button = gtk_button_new_from_stock (GTK_STOCK_CLOSE);
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                                        G_CALLBACK (gtk_widget_destroy), window);
gtk_widget_show (button);
gtk_widget_show_all (window);
}
void reset()
{
FILE *fp;
char http[120]="",ftp[120]="",pos[120]="",outhttp[120]="",outftp[120]="";
fp=fopen("save.dat","w");
fclose(fp);
fp=fopen("delin.dat","w");
fclose(fp);
fp=fopen("delout.dat","w");
fclose(fp);
fp=fopen("setting.dat","r");
while(!feof(fp))
     {
      fread(&x,sizeof x,1,fp);
     }
fclose(fp);
```

```
system("service iptables stop");
system("service iptables start");
strcat(pos,"iptables -t nat -A POSTROUTING -j SNAT --to-source  ");
strcat(pos,x.FPUBLIC);
strcat(pos," -s ");
strcat(pos,x.SUB);
strcat(http,"iptables -t nat -A PREROUTING -p tcp --dport 80 -d  ");
strcat(http,x.FPUBLIC);
strcat(http," -j DNAT  --to-destination  ");
strcat(http,x.HTTP);
strcat(ftp,"iptables -t nat -A PREROUTING -p tcp  --dport 21 -d  ");
strcat(ftp,x.FPUBLIC);
strcat(ftp," -j DNAT  --to-destination  ");
strcat(ftp,x.FTP);
strcat(outhttp,"iptables -t nat -A OUTPUT -p tcp --dport 80 -d  ");
strcat(outhttp,x.FPUBLIC);
strcat(outhttp," -j DNAT  --to-destination  ");
strcat(outhttp,x.HTTP);
strcat(outftp,"iptables -t nat -A OUTPUT -p tcp  --dport 21 -d  ");
strcat(outftp,x.FPUBLIC);
strcat(outftp," -j DNAT  --to-destination  ");
strcat(outftp,x.FTP);
system(ftp);
system(http);
system(outftp);
system(outhttp);
system(pos);
save();
prog();
}
void restore()
{
system("iptables-restore save.dat");
}
void save()
{
system("iptables-save -c >save.dat");
}
void add_out_rule()
{
/* GtkWidget is the storage type for widgets */
GtkWidget *window;
GtkWidget *frame;
GtkWidget *entry;
GtkWidget *entry1;
GtkWidget *vbox;
GtkWidget *hbox;
GtkWidget *box2;
GtkWidget *box1;
```

```
GtkWidget *main_box;
GtkWidget *button;
GtkWidget *button1;
GtkWidget *button2;
GtkWidget *button3;
GtkWidget *button4;
GtkWidget *button5;
GtkWidget *button6;
GtkWidget *button7;
GtkWidget *label;
GSList *group;
GSList *group1;
gint tmp_pos;
window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
gtk_window_set_title (GTK_WINDOW (window), "Adding Outgoing Rule");
gtk_widget_set_size_request (window, 350, 380);
gtk_window_set_position (window, GTK_WIN_POS_CENTER_ALWAYS);
gtk_window_set_resizable(window,FALSE);
/* Sets the border width of the window. */
gtk_container_set_border_width (GTK_CONTAINER (window), 10);
/* Create a Frame */
frame = gtk_frame_new (NULL);
gtk_container_add (GTK_CONTAINER (window), frame);
gtk_box_pack_start (GTK_BOX (window), frame, FALSE, FALSE, 10);
gtk_widget_show (frame);
/*Show Enery Boxes*/
main_box = gtk_vbox_new (FALSE, 0);
gtk_container_add (GTK_CONTAINER (frame), main_box);
gtk_container_set_border_width (GTK_CONTAINER (main_box), 10);
gtk_box_pack_start (GTK_BOX (frame), main_box, FALSE, FALSE, 10);
gtk_widget_show (main_box);
hbox = gtk_hbox_new (TRUE, 0);
label = gtk_label_new ("Source IP Address:        Distenation IP Address:");
gtk_container_add (GTK_CONTAINER (hbox), label);
gtk_container_set_border_width (GTK_CONTAINER (hbox), 10);
hbox = gtk_hbox_new (FALSE, 10);
gtk_container_add (GTK_CONTAINER (main_box), hbox);
gtk_container_set_border_width (GTK_CONTAINER (hbox), 10);
gtk_box_pack_start (GTK_BOX (main_box), hbox, FALSE, FALSE, 0);
gtk_widget_show (hbox);
vbox = gtk_vbox_new (FALSE, 0);
gtk_container_add (GTK_CONTAINER (hbox), vbox);
gtk_box_pack_start (GTK_BOX (hbox), vbox, FALSE, FALSE, 0);
gtk_widget_show (vbox);
entry = gtk_entry_new ();
gtk_entry_set_max_length (GTK_ENTRY (entry), 20);
gtk_entry_set_text (GTK_ENTRY (entry), "Sour. IP");
tmp_pos = GTK_ENTRY (entry)->text_length;
gtk_editable_insert_text (GTK_EDITABLE (entry), " Address", -1, &tmp_pos);
```

```
gtk_editable_select_region (GTK_EDITABLE (entry),0, GTK_ENTRY
(entry)->text_length);
gtk_box_pack_start (GTK_BOX (vbox), entry, TRUE, TRUE,0);
gtk_widget_show (entry);
vbox = gtk_vbox_new (FALSE, 0);
gtk_container_add (GTK_CONTAINER (hbox), vbox);
gtk_widget_show (vbox);
entry1 = gtk_entry_new ();
gtk_entry_set_max_length (GTK_ENTRY (entry1), 20);
gtk_entry_set_text (GTK_ENTRY (entry1), "Dist. IP ");
tmp_pos = GTK_ENTRY (entry1)->text_length;
gtk_editable_insert_text (GTK_EDITABLE (entry1), "Address", -1, &tmp_pos);
gtk_editable_select_region (GTK_EDITABLE (entry1),0, GTK_ENTRY
(entry1)->text_length);
gtk_box_pack_start (GTK_BOX (vbox), entry1, TRUE, TRUE,0);
gtk_widget_show (entry1);
/*Create Protocols Radiobutton Box*/
box1 = gtk_hbox_new (FALSE, 10);
gtk_container_add (GTK_CONTAINER (main_box), box1);
gtk_container_set_border_width (GTK_CONTAINER (box1), 10);
gtk_box_pack_start (GTK_BOX (main_box), box1, FALSE, FALSE, 10);
gtk_widget_show (box1);
frame = gtk_frame_new ("Protocols");
gtk_container_add (GTK_CONTAINER (box1), frame);
gtk_box_pack_start (GTK_BOX (box1), frame, FALSE, FALSE, 10);
gtk_widget_show (frame);
box2 = gtk_vbox_new (FALSE, 10);
gtk_container_set_border_width (GTK_CONTAINER (box2), 10);
gtk_container_add (GTK_CONTAINER (frame), box2);
gtk_box_pack_start (GTK_BOX (frame), box2, TRUE, TRUE, 20);
gtk_widget_show (box2);
button = gtk_radio_button_new_with_label (NULL, "TCP");
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
g_signal_connect (G_OBJECT (button), "pressed",
                                                G_CALLBACK (act_tcp),NULL);
gtk_widget_show (button);
group = gtk_radio_button_get_group (GTK_RADIO_BUTTON (button));
button1 = gtk_radio_button_new_with_label (group, "UDP");
g_signal_connect (G_OBJECT (button1), "pressed",
                                                G_CALLBACK (act_udp),NULL);
gtk_box_pack_start (GTK_BOX (box2), button1, TRUE, TRUE, 0);
gtk_widget_show (button1);
button2 = gtk_radio_button_new_with_label_from_widget (
                                        GTK_RADIO_BUTTON (button),"ICMP");
gtk_box_pack_start (GTK_BOX (box2), button2, TRUE, TRUE, 0);
g_signal_connect (G_OBJECT (button2), "pressed",
                                                G_CALLBACK (act_icmp),NULL);
gtk_widget_show (button2);
button3 = gtk_radio_button_new_with_label_from_widget (
```

```
                                                GTK_RADIO_BUTTON (button),"ALL");
gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (button3), TRUE);
g_signal_connect (G_OBJECT (button3), "pressed",
                                                G_CALLBACK (act_all),NULL);
gtk_box_pack_start (GTK_BOX (box2), button3, TRUE, TRUE, 0);
gtk_widget_show (button3);
/*Create Polcy Button Box*/
frame = gtk_frame_new ("Policy");
gtk_container_add (GTK_CONTAINER (box1), frame);
gtk_box_pack_start (GTK_BOX (box1), frame, FALSE, FALSE, 0);
gtk_widget_show (frame);
box2 = gtk_vbox_new (FALSE, 10);
gtk_container_set_border_width (GTK_CONTAINER (box2), 5);
gtk_container_add (GTK_CONTAINER (frame), box2);
gtk_box_pack_start (GTK_BOX (frame), box2, TRUE, TRUE, 20);
gtk_widget_show (box2);
button4 = gtk_radio_button_new_with_label (NULL, "Accept");
gtk_box_pack_start (GTK_BOX (box2), button4, TRUE, TRUE, 0);
g_signal_connect (G_OBJECT (button4), "pressed",
                                              G_CALLBACK (act_accept),NULL);
gtk_widget_show (button4);
group1 = gtk_radio_button_get_group (GTK_RADIO_BUTTON (button4));
button5 = gtk_radio_button_new_with_label (group1, "Drop");
g_signal_connect (G_OBJECT (button5), "pressed",
                                               G_CALLBACK (act_drop),NULL);
gtk_box_pack_start (GTK_BOX (box2), button5, TRUE, TRUE, 0);
gtk_widget_show (button5);
button6 = gtk_radio_button_new_with_label_from_widget (
                                           GTK_RADIO_BUTTON (button4),"Reject");
gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (button6), TRUE);
g_signal_connect (G_OBJECT (button6), "pressed",
                                               G_CALLBACK (act_reject),NULL);
gtk_box_pack_start (GTK_BOX (box2), button6, TRUE, TRUE, 0);
gtk_widget_show (button6);
box2 = gtk_hbox_new (TRUE, 10);
gtk_container_set_border_width (GTK_CONTAINER (box2), 10);
gtk_box_pack_start (GTK_BOX (main_box), box2, FALSE, TRUE, 0);
gtk_widget_show (box2);
button7= gtk_button_new_from_stock (GTK_STOCK_APPLY);
gtk_box_pack_start (GTK_BOX (box2), button7, TRUE, TRUE, 0);
g_signal_connect (G_OBJECT (button7), "clicked",
                                                G_CALLBACK(get_soip),entry);
g_signal_connect (G_OBJECT (button7), "clicked",
                                                G_CALLBACK (get_dsip),entry1);
g_signal_connect (G_OBJECT (button7), "clicked",
                                               G_CALLBACK (comand1),NULL);
button = gtk_button_new_from_stock (GTK_STOCK_CANCEL);
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
GTK_WIDGET_SET_FLAGS (button, GTK_CAN_DEFAULT);
```

```
gtk_widget_grab_default (button);
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                                        G_CALLBACK (gtk_widget_destroy), window);
button = gtk_button_new_from_stock (GTK_STOCK_CLOSE);
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                                        G_CALLBACK (gtk_widget_destroy), window);
gtk_widget_show (button);
gtk_widget_show_all (window);
}
void del_in_rule( GtkWidget *widget,GtkWidget *entry )
{
const char *entry_text;
int de,k,i=1;
struct del delin,m,temp[100];
fp=fopen("delin.dat","r");
while(!feof(fp))
                {
                  fread(&m,sizeof m,1,fp);
                  temp[i]=m;
                     i++;
                }
fclose(fp);
entry_text=gtk_entry_get_text (GTK_ENTRY (entry));
de=atoi(entry_text);
temp[de].no=0;
delin=temp[de];
system(delin.in);
system(delin.flog);
system(delin.fin);
fp=fopen("delin.dat","w");
fclose(fp);
fp=fopen("delin.dat","a");
for(k=1;k<i-1;k++)
             {
              if(temp[k].no!=1)
                    continue;
              else
              fwrite(&temp[k],sizeof temp[k],1,fp);
             }
fclose(fp);
save();
prog();
}
void del_out_rule( GtkWidget *widget,GtkWidget *entry )
{
const char *entry_text;
int de,k,i=1;
struct del delin,m,temp[100];
```

```
fp=fopen("delout.dat","r");
while(!feof(fp))
                {
                  fread(&m,sizeof m,1,fp);
                  temp[i]=m;
                     i++;
                }
 fclose(fp);
 entry_text=gtk_entry_get_text (GTK_ENTRY (entry));
 de=atoi(entry_text);
 temp[de].no=0;
 delin=temp[de];
 system(delin.in);
 system(delin.flog);
 system(delin.fin);
 fp=fopen("delout.dat","w");
 fclose(fp);
 fp=fopen("delout.dat","a");
 for(k=1;k<i-1;k++)
             {
              if(temp[k].no!=1)
                    continue;
              else
              fwrite(&temp[k],sizeof temp[k],1,fp);
             }
fclose(fp);
save();
prog();
}
void remove_in_rule()
{
GtkWidget *window;
GtkWidget *frame;
GtkWidget *button;
GtkWidget *hbox;
GtkWidget *main_box;
GtkWidget *vbox;
GtkWidget *box2;
GtkWidget *entry;
GtkWidget *label;
gint tmp_pos;
window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
gtk_window_set_title (GTK_WINDOW (window), "Deleting Incomming Rule");
gtk_widget_set_size_request (window, 280, 180);
gtk_container_set_border_width (GTK_CONTAINER (window), 10);
gtk_window_set_position (window, GTK_WIN_POS_CENTER_ALWAYS);
gtk_window_set_resizable(window,FALSE);
frame = gtk_frame_new (NULL);
gtk_container_add (GTK_CONTAINER (window), frame);
```

```
gtk_frame_set_shadow_type (GTK_FRAME (frame),
                                                GTK_SHADOW_ETCHED_OUT);
gtk_widget_show (frame);
main_box = gtk_vbox_new (FALSE, 0);
gtk_container_add (GTK_CONTAINER (frame), main_box);
gtk_container_set_border_width (GTK_CONTAINER (main_box), 10);
gtk_box_pack_start (GTK_BOX (frame), main_box, FALSE, FALSE, 10);
gtk_widget_show (main_box);
hbox = gtk_hbox_new (TRUE, 0);
label = gtk_label_new ("The No of Deleated Rule:");
gtk_container_add (GTK_CONTAINER (hbox), label);
gtk_container_set_border_width (GTK_CONTAINER (hbox), 0);
gtk_box_pack_start (GTK_BOX (main_box), hbox, FALSE,FALSE,0);
hbox = gtk_hbox_new (FALSE, 0);
gtk_container_add (GTK_CONTAINER (main_box), hbox);
gtk_container_set_border_width (GTK_CONTAINER (hbox), 10);
gtk_box_pack_start (GTK_BOX (main_box), hbox, FALSE, FALSE, 0);
gtk_widget_show (hbox);
vbox = gtk_vbox_new (FALSE, 10);
gtk_container_add (GTK_CONTAINER (hbox), vbox);
gtk_box_pack_start (GTK_BOX (hbox), vbox, FALSE, FALSE, 0);
gtk_widget_show (vbox);
entry = gtk_entry_new ();
gtk_entry_set_max_length (GTK_ENTRY (entry), 40);
gtk_entry_set_text (GTK_ENTRY (entry), "Enter The Deleted");
tmp_pos = GTK_ENTRY (entry)->text_length;
gtk_editable_insert_text (GTK_EDITABLE (entry), " Rule No", -1, &tmp_pos);
gtk_editable_select_region (GTK_EDITABLE (entry),0, GTK_ENTRY
(entry)->text_length);
gtk_box_pack_start (GTK_BOX (vbox), entry, TRUE, TRUE,0);
gtk_widget_show (entry);
box2 = gtk_hbox_new (TRUE, 10);
gtk_container_set_border_width (GTK_CONTAINER (box2), 10);
gtk_container_add (GTK_CONTAINER (main_box), box2);
gtk_box_pack_start (GTK_BOX (main_box), box2, TRUE, TRUE, 20);
gtk_widget_show (box2);
button= gtk_button_new_from_stock (GTK_STOCK_APPLY);
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
g_signal_connect (G_OBJECT (button), "clicked",
                                                G_CALLBACK(del_in_rule),entry);
gtk_widget_show (button);
button = gtk_button_new_from_stock (GTK_STOCK_CANCEL);
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
GTK_WIDGET_SET_FLAGS (button, GTK_CAN_DEFAULT);
gtk_widget_grab_default (button);
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                                        G_CALLBACK (gtk_widget_destroy), window);
button= gtk_button_new_from_stock (GTK_STOCK_CLOSE);
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
```

```
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                                        G_CALLBACK (gtk_widget_destroy), window);
gtk_widget_show (button);
button = gtk_button_new_with_label ("View all Rules");
gtk_box_pack_start (GTK_BOX (vbox), button, TRUE, TRUE, 0);
g_signal_connect (G_OBJECT (button), "clicked",
                                                G_CALLBACK (view_all_in),NULL);
gtk_widget_show_all (window);
}
/*View Log File*/
static void menuitem_response( )
{
system("dmesg | grep Firewall > log.txt");
system("cat heder_view_log >view_log_table|cat log.txt>> view_log_table|gedit
view_log_table ");
}
static void view_all_in()
{
system("cat heder_in > incoming_rule_table|iptables -L INPUT  -n --line-numbers
|cat>>incoming_rule_table|gedit incoming_rule_table");
}
static void view_all_out()
{
system("cat heder_out > outgoing_rule_table | iptables --line-numbers -n -L
OUTPUT | cat >> outgoing_rule_table | gedit outgoing_rule_table");
}
void helpabout( )
{
GtkWidget *window;
GtkWidget *frame;
GtkWidget *label;
window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
gtk_window_set_title (GTK_WINDOW (window), "About Firewall Program");
gtk_widget_set_size_request (window, 300, 200);
gtk_window_set_position (window, GTK_WIN_POS_CENTER_ALWAYS);
gtk_window_set_resizable(window,FALSE);
gtk_container_set_border_width (GTK_CONTAINER (window), 10);
frame = gtk_frame_new (NULL);
gtk_container_add (GTK_CONTAINER (window), frame);
gtk_box_pack_start (GTK_BOX (window), frame, FALSE, FALSE, 10);
gtk_widget_show (frame);
label = gtk_label_new ("\nSudan University of Science and Technolog\n\tCollege of
Post Graduate Studies\n\n\n\t    This is Firewall Program \n\n      Created by:  Isam
Abdelnabi\n      Supervised by: Dr. Yahia Abdalla\n\n\t\t\t July 2005");
gtk_container_add (GTK_CONTAINER (frame), label);
gtk_box_pack_start (GTK_BOX (frame), label, FALSE, FALSE, 0);
gtk_widget_show (label);
gtk_widget_show (window);
}
```

```
void remove_out_rule()
{
GtkWidget *window;
GtkWidget *frame;
GtkWidget *button;
GtkWidget *hbox;
GtkWidget *main_box;
GtkWidget *vbox;
GtkWidget *box2;
GtkWidget *entry;
GtkWidget *label;
gint tmp_pos;
window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
gtk_window_set_title (GTK_WINDOW (window), "Deleting Outgoing Rule");
gtk_widget_set_size_request (window, 280, 180);
gtk_container_set_border_width (GTK_CONTAINER (window), 10);
gtk_window_set_position (window, GTK_WIN_POS_CENTER_ALWAYS);
gtk_window_set_resizable(window,FALSE);
frame = gtk_frame_new (NULL);
gtk_container_add (GTK_CONTAINER (window), frame);
gtk_frame_set_shadow_type (GTK_FRAME (frame),
                                                GTK_SHADOW_ETCHED_OUT);
gtk_widget_show (frame);
main_box = gtk_vbox_new (FALSE, 0);
gtk_container_add (GTK_CONTAINER (frame), main_box);
gtk_container_set_border_width (GTK_CONTAINER (main_box), 10);
gtk_box_pack_start (GTK_BOX (frame), main_box, FALSE, FALSE, 10);
gtk_widget_show (main_box);
hbox = gtk_hbox_new (TRUE, 0);
label = gtk_label_new ("The No of Deleated Rule:");
gtk_container_add (GTK_CONTAINER (hbox), label);
gtk_container_set_border_width (GTK_CONTAINER (hbox), 0);
gtk_box_pack_start (GTK_BOX (main_box), hbox, FALSE,FALSE,0);
hbox = gtk_hbox_new (FALSE, 0);
gtk_container_add (GTK_CONTAINER (main_box), hbox);
gtk_container_set_border_width (GTK_CONTAINER (hbox), 10);
gtk_box_pack_start (GTK_BOX (main_box), hbox, FALSE, FALSE, 0);
gtk_widget_show (hbox);
vbox = gtk_vbox_new (FALSE, 10);
gtk_container_add (GTK_CONTAINER (hbox), vbox);
gtk_box_pack_start (GTK_BOX (hbox), vbox, FALSE, FALSE, 0);
gtk_widget_show (vbox);
entry = gtk_entry_new ();
gtk_entry_set_max_length (GTK_ENTRY (entry), 40);
gtk_entry_set_text (GTK_ENTRY (entry), "Enter The Deleted");
tmp_pos = GTK_ENTRY (entry)->text_length;
gtk_editable_insert_text (GTK_EDITABLE (entry), " Rule No", -1, &tmp_pos);
gtk_editable_select_region (GTK_EDITABLE (entry),0, GTK_ENTRY
(entry)->text_length);
```

```
gtk_box_pack_start (GTK_BOX (vbox), entry, TRUE, TRUE,0);
gtk_widget_show (entry);
box2 = gtk_hbox_new (TRUE, 10);
gtk_container_set_border_width (GTK_CONTAINER (box2), 10);
gtk_container_add (GTK_CONTAINER (main_box), box2);
gtk_box_pack_start (GTK_BOX (main_box), box2, TRUE, TRUE, 20);
gtk_widget_show (box2);
button= gtk_button_new_from_stock (GTK_STOCK_APPLY);
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
g_signal_connect (G_OBJECT (button), "clicked",
                                                G_CALLBACK(del_out_rule),entry);
gtk_widget_show (button);
button = gtk_button_new_from_stock (GTK_STOCK_CANCEL);
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
GTK_WIDGET_SET_FLAGS (button, GTK_CAN_DEFAULT);
gtk_widget_grab_default (button);
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                                        G_CALLBACK (gtk_widget_destroy), window);
button= gtk_button_new_from_stock (GTK_STOCK_CLOSE);
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                                        G_CALLBACK (gtk_widget_destroy), window);
gtk_widget_show (button);
button = gtk_button_new_with_label ("View all Rules");
gtk_box_pack_start (GTK_BOX (vbox), button, TRUE, TRUE, 0);
g_signal_connect (G_OBJECT (button), "clicked", G_CALLBACK (
                                                        view_all_out),NULL);
gtk_widget_show_all (window);
}
/* Clean up allocated memory and remove the timer */
void destroy_progress( GtkWidget *widget,ProgressData *pdata)
{
gtk_timeout_remove (pdata->timer);
pdata->timer = 0;
pdata->window = NULL;
g_free (pdata);
gtk_main_quit ();
}
/* Update the value of the progress bar so that we get some movement */
gint progress_timeout( gpointer data )
{
ProgressData *pdata = (ProgressData *)data;
gdouble new_val;
pdata->activity_mode=FALSE;
if (pdata->activity_mode)
gtk_progress_bar_pulse (GTK_PROGRESS_BAR (pdata->pbar));
else
{
```

```
/* Calculate the value of the progress bar using the value range set in the adjustment
object */
 new_val = gtk_progress_bar_get_fraction (GTK_PROGRESS_BAR
(pdata->pbar)) + 0.01;
if (new_val > 1.0)
{
buttonp=gtk_button_new_from_stock (GTK_STOCK_CLOSE);
gtk_box_pack_start (GTK_BOX (vboxp), buttonp, TRUE, FALSE,0);
g_signal_connect_swapped (G_OBJECT (buttonp), "clicked",
                                    G_CALLBACK (gtk_widget_destroy), pdata->window);
gtk_widget_show (buttonp);
return FALSE;
}
/* Set the new value */
gtk_progress_bar_set_fraction (GTK_PROGRESS_BAR (pdata->pbar), new_val);
}
/* As this is a timeout function, return TRUE so that it continues to get called */
return TRUE;
}
void prog()
{
ProgressData *pdata;
GtkWidget *align;
/* Allocate memory for the data that is passed to the callbacks */
pdata = g_malloc (sizeof (ProgressData));
pdata->window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
gtk_window_set_resizable (GTK_WINDOW (pdata->window), TRUE);
g_signal_connect (G_OBJECT (pdata->window), "destroy",
                                              G_CALLBACK (destroy_progress),pdata);
gtk_window_set_position (pdata->window,
                                                GTK_WIN_POS_CENTER_ALWAYS);
gtk_window_set_resizable(pdata->window,FALSE);
gtk_window_set_title (GTK_WINDOW (pdata->window), "System Udate ...");
gtk_container_set_border_width (GTK_CONTAINER (pdata->window), 0);
vboxp = gtk_vbox_new (FALSE, 5);
gtk_container_set_border_width (GTK_CONTAINER (vboxp), 10);
gtk_container_add (GTK_CONTAINER (pdata->window), vboxp);
gtk_widget_show (vboxp);
/* Create a centering alignment object */
align = gtk_alignment_new (0.5, 0.5, 0, 0);
gtk_box_pack_start (GTK_BOX (vboxp), align, FALSE, FALSE, 5);
gtk_widget_show (align);
/* Create the GtkProgressBar */
pdata->pbar = gtk_progress_bar_new ();
gtk_container_add (GTK_CONTAINER (align), pdata->pbar);
gtk_widget_show (pdata->pbar);
pdata->activity_mode = !pdata->activity_mode;
/* Add a timer callback to update the value of the progress bar */
pdata->timer = gtk_timeout_add (30, progress_timeout, pdata);
```

```
gtk_widget_show (pdata->window);
gtk_main ();
}
void view_backup()
{
system("cat heder_view_log_backup >view_log_table_backup|cat  gedit
Backup_of_log_file.txt>> view_log_table_backup|gedit view_log_table_backup ");
}
void reset_backup()
{
FILE *fp;
fp=fopen("Backup_of_log_file.txt","w");
fclose(fp);
prog();
}
void getFPUBLIC( GtkWidget *widget,GtkWidget *entry )
{
struct hostent *ptrh;
const char *entry_text1;
entry_text1= gtk_entry_get_text (GTK_ENTRY (entry));
strcpy(x.FPUBLIC,entry_text1);
ptrh = gethostbyname(x.FPUBLIC);
if(((char*)ptrh) == NULL)
      {
       error_fpip();
      }
}
void getFLOCAL( GtkWidget *widget,GtkWidget *entry )
{
struct hostent *ptrh;
const char *entry_text2;
entry_text2 = gtk_entry_get_text (GTK_ENTRY (entry));
strcpy(x.FLOCAL,entry_text2);
ptrh = gethostbyname(x.FLOCAL);
if(((char*)ptrh) == NULL)
       {
        error_flip();
       }
}
void apply()
{
char http[120]="", ftp[120]="",pos[120]="";
char outhttp[120]="", outftp[120]="";
struct hostent *ptrh1,*ptrh2,*ptrh3,*ptrh4;
ptrh1 = gethostbyname(x.FPUBLIC);
ptrh2 = gethostbyname(x.FLOCAL);
ptrh3 = gethostbyname(x.FTP);
ptrh4 = gethostbyname(x.HTTP);
```

```
if((((char*)ptrh1) == NULL)||(((char*)ptrh2) == NULL)||(((char*)ptrh3) ==
NULL)||(((char*)ptrh4) == NULL))
printf("");
else
{
system("iptables -t nat -D PREROUTING 2");
system("iptables -t nat -D PREROUTING  1");
system("iptables -t nat -D OUTPUT 2");
system("iptables -t nat -D OUTPUT  1");
system("iptables -t nat -D POSTROUTING  1");
strcat(pos,"iptables -t nat -A POSTROUTING -j SNAT --to-source  ");
strcat(pos,x.FPUBLIC);
strcat(pos," -s ");
strcat(pos,x.SUB);
strcat(http,"iptables -t nat -A PREROUTING -p tcp --dport 80 -d  ");
strcat(http,x.FPUBLIC);
strcat(http," -j DNAT  --to-destination  ");
strcat(http,x.HTTP);
strcat(outhttp,"iptables -t nat -A OUTPUT -p tcp --dport 80 -d  ");
strcat(outhttp,x.FPUBLIC);
strcat(outhttp," -j DNAT  --to-destination  ");
strcat(outhttp,x.HTTP);
strcat(ftp,"iptables -t nat -A PREROUTING -p tcp  --dport 21 -d  ");
strcat(ftp,x.FPUBLIC);
strcat(ftp," -j DNAT  --to-destination  ");
strcat(ftp,x.FTP);
strcat(outftp,"iptables -t nat -A OUTPUT -p tcp  --dport 21 -d  ");
strcat(outftp,x.FPUBLIC);
strcat(outftp," -j DNAT  --to-destination  ");
strcat(outftp,x.FTP);
system(pos);
system(ftp);
system(http);
system(outftp);
system(outhttp);
prog();
fp=fopen("setting.dat","w");
fwrite(&x,sizeof x,1,fp);
fclose(fp);
save();
}
}
void getFTP( GtkWidget *widget,GtkWidget *entry )
{
struct hostent *ptrh;
const char *entry_text3;
entry_text3 = gtk_entry_get_text (GTK_ENTRY (entry));
strcpy(x.FTP,entry_text3);
ptrh = gethostbyname(x.FTP);
```

```
if(((char*)ptrh) == NULL)
      {
       error_ftp();
      }
}
void getHTTP( GtkWidget *widget,GtkWidget *entry )
{
struct hostent *ptrh;
const char *entry_text4;
entry_text4 = gtk_entry_get_text (GTK_ENTRY (entry));
strcpy(x.HTTP,entry_text4);
ptrh = gethostbyname(x.HTTP);
if(((char*)ptrh) == NULL)
      {
       error_http();
      }
}
void getSUB( GtkWidget *widget,GtkWidget *entry )
{
const char *entry_text5;
entry_text5 = gtk_entry_get_text (GTK_ENTRY (entry));
strcpy(x.SUB,entry_text5);
}
void error_fpip()
{
GtkWidget *dialog1;
GtkWidget *window1;
char text[50]=("Error Invalid Firewall Public Address: ");
strcat(text,x.FPUBLIC);
window1 = gtk_window_new (GTK_WINDOW_TOPLEVEL);
dialog1 = gtk_message_dialog_new (window1,
 GTK_DIALOG_DESTROY_WITH_PARENT,GTK_MESSAGE_ERROR,GTK_B
                                                    UTTONS_CLOSE,text);
gtk_window_set_position (dialog1, GTK_WIN_POS_CENTER_ALWAYS);
gtk_dialog_run (GTK_DIALOG (dialog1));
gtk_widget_destroy (dialog1);
}
void error_flip()
{
GtkWidget *dialog1;
GtkWidget *window1;
char text[50]=("Error Invalid Firewall Local Address: ");
strcat(text,x.FLOCAL);
window1 = gtk_window_new (GTK_WINDOW_TOPLEVEL);
dialog1 = gtk_message_dialog_new (window1,
 GTK_DIALOG_DESTROY_WITH_PARENT,GTK_MESSAGE_ERROR,GTK_B
                                                    UTTONS_CLOSE,text);
gtk_window_set_position (dialog1, GTK_WIN_POS_CENTER_ALWAYS);
gtk_dialog_run (GTK_DIALOG (dialog1));
```

```
gtk_widget_destroy (dialog1);
}
void error_http()
{
GtkWidget *dialog1;
GtkWidget *window1;
char text[50]=("Error Invalid HTTP Address: ");
strcat(text,x.HTTP);
window1 = gtk_window_new (GTK_WINDOW_TOPLEVEL);
dialog1 = gtk_message_dialog_new (window1,
 GTK_DIALOG_DESTROY_WITH_PARENT,GTK_MESSAGE_ERROR,GTK_B
                                                    UTTONS_CLOSE,text);
gtk_window_set_position (dialog1, GTK_WIN_POS_CENTER_ALWAYS);
gtk_dialog_run (GTK_DIALOG (dialog1));
gtk_widget_destroy (dialog1);
}
void error_ftp()
{
GtkWidget *dialog1;
GtkWidget *window1;
char text[50]=("Error Invalid FTP Address: ");
strcat(text,x.FTP);
window1 = gtk_window_new (GTK_WINDOW_TOPLEVEL);
dialog1 = gtk_message_dialog_new (window1,
 GTK_DIALOG_DESTROY_WITH_PARENT,GTK_MESSAGE_ERROR,GTK_B
                                                    UTTONS_CLOSE,text);
gtk_window_set_position (dialog1, GTK_WIN_POS_CENTER_ALWAYS);
gtk_dialog_run (GTK_DIALOG (dialog1));
gtk_widget_destroy (dialog1);
}
void entry_toggle_editable( GtkWidget *checkbutton,GtkWidget *entry )
{
gtk_editable_set_editable (GTK_EDITABLE (entry),
                                    GTK_TOGGLE_BUTTON (checkbutton)->active);
}
void setting()
{
GtkWidget *window;
GtkWidget *vbox;
GtkWidget *hbox;
GtkWidget *hbox1;
GtkWidget *hbox2;
GtkWidget *main_vbox;
GtkWidget *entry,*entry1,*entry2,*entry3,*entry4;
GtkWidget *button;
GtkWidget *frame;
GtkWidget *label;
GtkWidget *check;
struct ipaddresses  s;
```

```
fp=fopen("setting.dat","r");
while(!feof(fp))
{
fread(&s,sizeof s,1,fp);
}
fclose(fp);
/* create a new window */
window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
gtk_widget_set_size_request (GTK_WIDGET (window), 300, 270);
gtk_window_set_position (window, GTK_WIN_POS_CENTER_ALWAYS);
gtk_window_set_resizable(window,FALSE);
gtk_window_set_title (GTK_WINDOW (window), "Firewall Setting");
g_signal_connect (G_OBJECT (window), "destroy",
                                        G_CALLBACK (gtk_widget_destroy), window);
frame = gtk_frame_new (NULL);
gtk_container_add (GTK_CONTAINER (window), frame);
gtk_container_set_border_width (GTK_CONTAINER (frame), 10);
/*Create Main Box to hold all other forms*/
main_vbox = gtk_vbox_new (FALSE,0);
gtk_container_add (GTK_CONTAINER (frame), main_vbox);
gtk_container_set_border_width (GTK_CONTAINER (main_vbox), 10);
hbox = gtk_hbox_new (TRUE, 20);
gtk_container_add (GTK_CONTAINER (main_vbox), hbox);
gtk_container_set_border_width (GTK_CONTAINER (main_vbox), 0);
gtk_widget_show (hbox);
vbox = gtk_vbox_new (FALSE, 10);
gtk_container_add (GTK_CONTAINER (hbox), vbox);
gtk_container_set_border_width (GTK_CONTAINER (hbox), 10);
gtk_widget_show (vbox);
label = gtk_label_new ("Firewall Public IP:     ");
gtk_box_pack_start (GTK_BOX (vbox), label, TRUE, TRUE, 0);
entry = gtk_entry_new ();
gtk_entry_set_max_length (GTK_ENTRY (entry), 50);
g_signal_connect (G_OBJECT (entry), "activate",
                                            G_CALLBACK (getFPUBLIC),entry);
gtk_entry_set_text (GTK_ENTRY (entry),s.FPUBLIC);
gtk_entry_set_visibility (GTK_ENTRY (entry),TRUE);
gtk_editable_set_editable(GTK_ENTRY (entry),FALSE);
gtk_box_pack_start (GTK_BOX (vbox), entry, TRUE, TRUE, 0);
gtk_widget_show (entry);
label = gtk_label_new ("Firewall Local IP:     ");
gtk_box_pack_start (GTK_BOX (vbox), label, TRUE, TRUE, 0);
entry1 = gtk_entry_new ();
gtk_entry_set_max_length (GTK_ENTRY (entry1), 50);
g_signal_connect (G_OBJECT (entry1), "activate",
                                            G_CALLBACK (getFLOCAL),entry1);
gtk_editable_set_editable(GTK_ENTRY (entry1),FALSE);
gtk_entry_set_text (GTK_ENTRY (entry1),s.FLOCAL);
gtk_entry_set_visibility (GTK_ENTRY (entry1),TRUE);
```

```
gtk_box_pack_start (GTK_BOX (vbox), entry1, TRUE, TRUE, 0);
gtk_widget_show (entry1);
label = gtk_label_new ("Subnet Add./mask:");
gtk_box_pack_start (GTK_BOX (vbox), label, TRUE, TRUE, 0);
entry4 = gtk_entry_new ();
gtk_entry_set_max_length (GTK_ENTRY (entry4), 50);
g_signal_connect (G_OBJECT (entry4), "activate",
                                                    G_CALLBACK (getSUB),entry4);
gtk_editable_set_editable(GTK_ENTRY (entry4),FALSE);
gtk_entry_set_text (GTK_ENTRY (entry4), s.SUB);
gtk_entry_set_visibility (GTK_ENTRY (entry4),TRUE);
gtk_box_pack_start (GTK_BOX (vbox), entry4, TRUE, TRUE,0);
gtk_widget_show (entry4);
vbox = gtk_vbox_new (FALSE, 10);
gtk_container_add (GTK_CONTAINER (hbox), vbox);
gtk_container_set_border_width (GTK_CONTAINER (hbox), 10);
gtk_widget_show (vbox);
label = gtk_label_new ("HTTP Server IP:      ");
gtk_box_pack_start (GTK_BOX (vbox), label, TRUE, TRUE, 0);
entry2 = gtk_entry_new ();
gtk_entry_set_max_length (GTK_ENTRY (entry2), 50);
g_signal_connect (G_OBJECT (entry2), "activate",
                                                  G_CALLBACK (getHTTP),entry2);
gtk_editable_set_editable(GTK_ENTRY (entry2),FALSE);
gtk_entry_set_text (GTK_ENTRY (entry2), s.HTTP);
gtk_entry_set_visibility (GTK_ENTRY (entry2),TRUE);
gtk_box_pack_start (GTK_BOX (vbox), entry2, TRUE, TRUE, 0);
gtk_widget_show (entry2);
label = gtk_label_new ("FTP Server IP:\t      ");
gtk_box_pack_start (GTK_BOX (vbox), label, TRUE, TRUE, 0);
entry3 = gtk_entry_new ();
gtk_entry_set_max_length (GTK_ENTRY (entry3), 50);
g_signal_connect (G_OBJECT (entry3), "activate",
                                                    G_CALLBACK (getFTP),entry3);
gtk_editable_set_editable(GTK_ENTRY (entry3),FALSE);
gtk_entry_set_text (GTK_ENTRY (entry3), s.FTP);
gtk_entry_set_visibility (GTK_ENTRY (entry3),TRUE);
gtk_box_pack_start (GTK_BOX (vbox), entry3, TRUE, TRUE, 0);
gtk_widget_show (entry3);
hbox2 = gtk_hbox_new (FALSE, 0);
gtk_box_pack_start (GTK_BOX (main_vbox), hbox2, TRUE, TRUE,10);
gtk_container_add (GTK_CONTAINER (main_vbox), hbox2);
gtk_widget_show (hbox2);
label = gtk_label_new (" ");
gtk_box_pack_start (GTK_BOX (vbox), label, TRUE, TRUE, 0);
check = gtk_check_button_new_with_label ("Change Setting");
gtk_box_pack_start (GTK_BOX (vbox), check, TRUE, TRUE,0);
gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (check), FALSE);
g_signal_connect (G_OBJECT (check), "toggled",
```

```
                                        G_CALLBACK (entry_toggle_editable), entry);
g_signal_connect (G_OBJECT (check), "toggled",
                                        G_CALLBACK (entry_toggle_editable), entry1);
g_signal_connect (G_OBJECT (check), "toggled",
                                        G_CALLBACK (entry_toggle_editable), entry2);
g_signal_connect (G_OBJECT (check), "toggled",
                                        G_CALLBACK (entry_toggle_editable), entry3);
g_signal_connect (G_OBJECT (check), "toggled",
                                        G_CALLBACK (entry_toggle_editable), entry4);
gtk_widget_show (check);
hbox1 = gtk_hbox_new (FALSE,0);
gtk_box_pack_start (GTK_BOX (main_vbox), hbox1, TRUE, TRUE,10);
gtk_container_add (GTK_CONTAINER (main_vbox), hbox1);
gtk_widget_show (hbox1);
button = gtk_button_new_from_stock (GTK_STOCK_OK);
gtk_box_pack_start(GTK_BOX (hbox1), button, TRUE, TRUE,10);
g_signal_connect (G_OBJECT (button), "clicked", G_CALLBACK(apply),NULL);
g_signal_connect (G_OBJECT (button), "clicked",
                                          G_CALLBACK (getFPUBLIC),entry);
g_signal_connect (G_OBJECT (button), "clicked",
                                         G_CALLBACK (getFLOCAL),entry1);
g_signal_connect (G_OBJECT (button), "clicked",
                                            G_CALLBACK (getHTTP),entry2);
g_signal_connect(G_OBJECT (button), "clicked",
                                             G_CALLBACK (getFTP),entry3);
g_signal_connect(G_OBJECT (button), "clicked",
                                            G_CALLBACK (getSUB),entry4);
gtk_widget_show (button);
button = gtk_button_new_from_stock (GTK_STOCK_CLOSE);
g_signal_connect_swapped(G_OBJECT (button),"clicked",
                                        G_CALLBACK(gtk_widget_destroy),window);
gtk_box_pack_start (GTK_BOX (hbox1), button, TRUE, TRUE,10);
GTK_WIDGET_SET_FLAGS (button, GTK_CAN_DEFAULT);
gtk_widget_grab_default (button);
gtk_widget_show (button);
gtk_widget_show_all (window);
}
viod help()
{
system("gedit  help.txt ");
}
```

Appendix B: Copy of Incoming Rules

Sudan University of Science and Technology
Firewall Program
Incoming Rules Table

num	target	prot	source	destination
1	REJECT	tcp	64.1.39.2.30	192.168.2.30 tcp dpt:21
2	REJECT	tcp	64.1.39.2.30	192.168.2.28 tcp dpt:80
3	ACCEPT	udp	212.0.139.1	192.168.2.30 tcp dpt:21
4	DROP	icmp	212.0.139.1	192.168.2.28 tcp dpt:80
5	REJECT	all	212.0.139.1	192.168.2.30 tcp dpt:21
6	DROP	udp	212.0.139.1	192.168.2.28 tcp dpt:80
7	REJECT	tcp	99.168.2.19	192.168.2.30 tcp dpt:21
8	ACCEPT	tcp	56.1.2.30	192.168.2.28 tcp dpt:80
9	REJECT	tcp	99.168.2.19	192.168.2.30 tcp dpt:21
10	ACCEPT	udp	99.168.2.19	192.168.2.28 tcp dpt:80
11	DROP	udp	56.1.2.30	192.168.2.30 tcp dpt:21
12	DROP	icmp	64.1.39.2.30	192.168.2.28 tcp dpt:80
13	REJECT	icmp	64.1.39.2.30	192.168.2.30 tcp dpt:21
14	ACCEPT	icmp	64.1.39.2.30	192.168.2.28 tcp dpt:80
15	ACCEPT	icmp	64.1.39.2.30	192.168.2.30 tcp dpt:21
16	REJECT	tcp	56.1.2.30	192.168.2.28 tcp dpt:80
17	REJECT	tcp	99.168.2.19	192.168.2.30 tcp dpt:21
18	ACCEPT	tcp	99.168.2.19	192.168.2.28 tcp dpt:80
19	REJECT	tcp	99.168.2.19	192.168.2.30 tcp dpt:21
20	REJECT	tcp	99.168.2.19	192.168.2.28 tcp dpt:80
21	REJECT	tcp	64.1.39.2.30	192.168.2.30 tcp dpt:21
22	REJECT	tcp	64.1.39.2.30	192.168.2.28 tcp dpt:80
23	REJECT	tcp	64.1.39.2.40	192.168.2.28 tcp dpt:80

Appendix C: Copy of Outgoing Rules

Sudan University of Science and Technology
Firewall Program
Outgoing Rules Table

num	target	prot	source	destination
6	REJECT	tcp	192.168.2.3	64.1.39.2.30
7	REJECT	tcp	192.168.2.2	64.1.39.2.30
8	ACCEPT	udp	192.168.2.20	212.0.139.1
9	DROP	icmp	192.168.2.4	212.0.139.1
10	REJECT	all	192.168.2.5	212.0.139.1
11	DROP	udp	192.168.2.7	212.0.139.1
12	REJECT	tcp	192.168.2.8	99.168.2.19
13	ACCEPT	tcp	192.168.2.1	56.1.2.30
14	REJECT	tcp	192.168.2.18	99.168.2.19
15	ACCEPT	udp	192.168.2.22	99.168.2.19
16	DROP	udp	192.168.2.24	56.1.2.30
17	DROP	icmp	192.168.2.13	64.1.39.2.30
18	REJECT	icmp	192.168.2.25	64.1.39.2.30
19	ACCEPT	icmp	192.168.2.31	64.1.39.2.30
20	ACCEPT	icmp	192.168.2.32	64.1.39.2.30
21	REJECT	tcp	192.168.2.11	56.1.2.30
22	REJECT	tcp	192.168.2.17	99.168.2.19
23	ACCEPT	tcp	192.168.2.43	99.168.2.19
24	REJECT	tcp	192.168.2.55	99.168.2.19
25	REJECT	tcp	192.168.2.56	99.168.2.19
26	REJECT	tcp	192.168.2.23	64.1.39.2.30
27	REJECT	tcp	192.168.2.22	64.1.39.2.30
28	REJECT	tcp	192.168.2.25	64.1.39.2.30
29	REJECT	tcp	192.168.2.24	64.1.39.2.30

Appendix D: Copy of Logging File

FirewallIN=eth0 OUT=eth1 SRC=212.1.1.32 DST=192.168.2.28 LEN=48 TOS=0x00 PREC=0x00 TTL=127 ID=809 DF PROTO=TCP SPT=1071 DPT=80 WINDOW=16384 RES=0x00 SYN URGP=0
FirewallIN=eth1 OUT=eth1 SRC=212.1.1.32 DST=192.168.2.28 LEN=48 TOS=0x00 PREC=0x00 TTL=127 ID=811 DF PROTO=TCP SPT=1071 DPT=80 WINDOW=16384 RES=0x00 SYN URGP=0
FirewallIN=eth0 OUT=eth1 SRC=212.1.1.32 DST=192.168.2.28 LEN=48 TOS=0x00 PREC=0x00 TTL=127 ID=812 DF PROTO=TCP SPT=1071 DPT=80 WINDOW=16384 RES=0x00 SYN URGP=0
FirewallIN=eth1 OUT=eth1 SRC=192.168.2.35 DST=192.168.2.28 LEN=48 TOS=0x00 PREC=0x00 TTL=127 ID=799 DF PROTO=TCP SPT=1103 DPT=80 WINDOW=16384 RES=0x00 SYN URGP=0
FirewallIN=eth1 OUT=eth1 SRC=192.168.2.35 DST=192.168.2.28 LEN=48 TOS=0x00 PREC=0x00 TTL=127 ID=801 DF PROTO=TCP SPT=1103 DPT=80 WINDOW=16384 RES=0x00 SYN URGP=0
FirewallIN=eth1 OUT=eth1 SRC=192.168.2.35 DST=192.168.2.28 LEN=48 TOS=0x00 PREC=0x00 TTL=127 ID=802 DF PROTO=TCP SPT=1103 DPT=80 WINDOW=16384 RES=0x00 SYN URGP=0

FirewallIN=eth0 OUT=eth1 SRC=212.1.1.50 DST=192.168.2.28 LEN=237
TOS=0x00 PREC=0x00 TTL=127 ID=887 DF PROTO=TCP SPT=1072 DPT=80
WINDOW=16844 RES=0x00 ACK PSH URGP=0
FirewallIN=eth0 OUT=eth1 SRC=212.1.1.50 DST=192.168.2.28 LEN=237
TOS=0x00 PREC=0x00 TTL=127 ID=889 DF PROTO=TCP SPT=1072 DPT=80
WINDOW=16844 RES=0x00 ACK PSH URGP=0
FirewallIN=eth0 OUT=eth1 SRC=212.1.1.50 DST=192.168.2.28 LEN=237
TOS=0x00 PREC=0x00 TTL=127 ID=890 DF PROTO=TCP SPT=1072 DPT=80
WINDOW=16844 RES=0x00 ACK PSH URGP=0
FirewallIN=eth0 OUT=eth1 SRC=212.1.1.50 DST=192.168.2.28 LEN=237
TOS=0x00 PREC=0x00 TTL=127 ID=891 DF PROTO=TCP SPT=1072 DPT=80
WINDOW=16844 RES=0x00 ACK PSH URGP=0
FirewallIN=eth0 OUT=eth1 SRC=212.1.1.50 DST=192.168.2.28 LEN=237
TOS=0x00 PREC=0x00 TTL=127 ID=892 DF PROTO=TCP SPT=1072 DPT=80
WINDOW=16844 RES=0x00 ACK PSH URGP=0
FirewallIN=eth0 OUT=eth1 SRC=212.1.1.50 DST=192.168.2.28 LEN=237
TOS=0x00 PREC=0x00 TTL=127 ID=894 DF PROTO=TCP SPT=1072 DPT=80
WINDOW=16844 RES=0x00 ACK PSH URGP=0
FirewallIN=eth0 OUT=eth1 SRC=212.1.1.50 DST=192.168.2.28 LEN=48
TOS=0x00 PREC=0x00 TTL=127 ID=896 DF PROTO=TCP SPT=1073 DPT=80
WINDOW=16384 RES=0x00 SYN URGP=0
FirewallIN=eth0 OUT=eth1 SRC=212.1.1.50 DST=192.168.2.28 LEN=48
TOS=0x00 PREC=0x00 TTL=127 ID=898 DF PROTO=TCP SPT=1073 DPT=80
WINDOW=16384 RES=0x00 SYN URGP=0
FirewallIN=eth0 OUT=eth1 SRC=212.1.1.50 DST=192.168.2.28 LEN=48
TOS=0x00 PREC=0x00 TTL=127 ID=899 DF PROTO=TCP SPT=1073 DPT=80
WINDOW=16384 RES=0x00 SYN URGP=0

Appendix E: Graphical User Interface

Figure E1 Main Graphical User Interface for Firewall

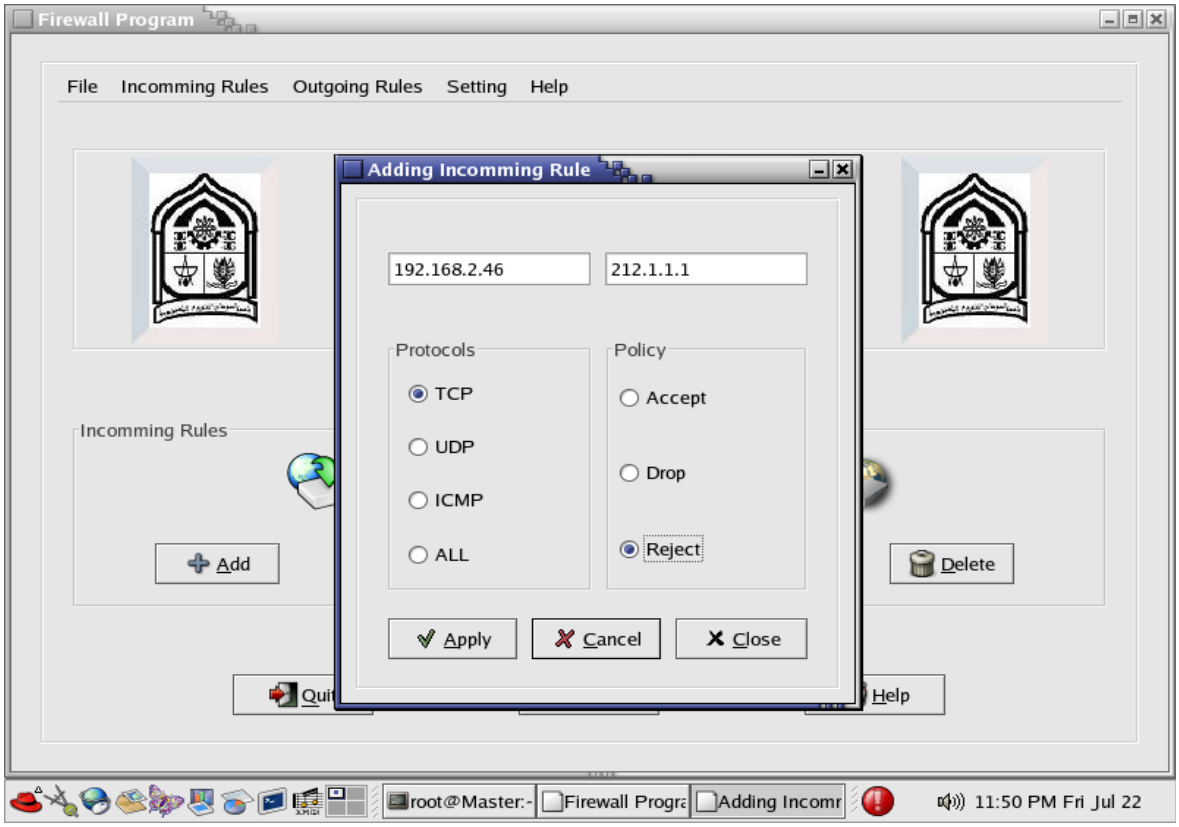


Figure E2 Graphical User Interface for Adding Incoming Rules.

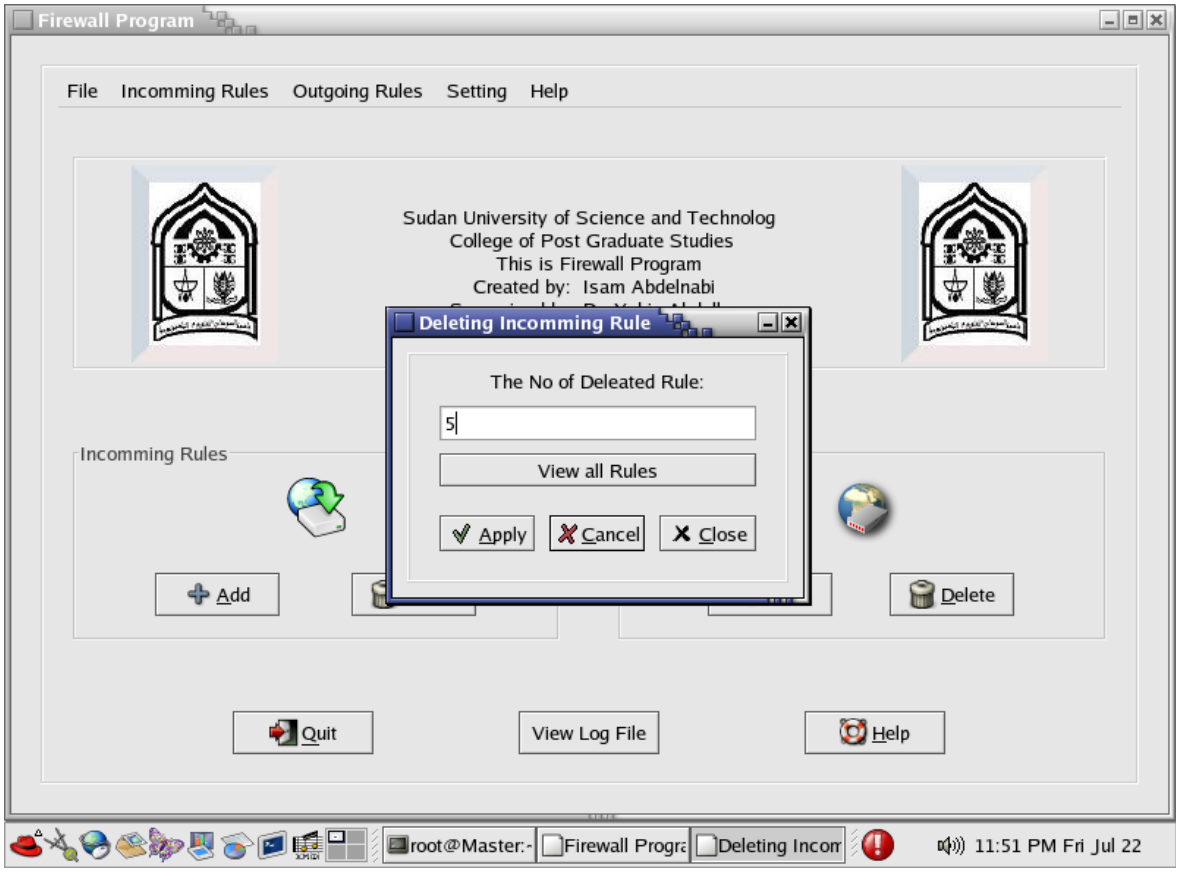


Figure E3 Graphical User Interface for Deleting Incoming Rules.

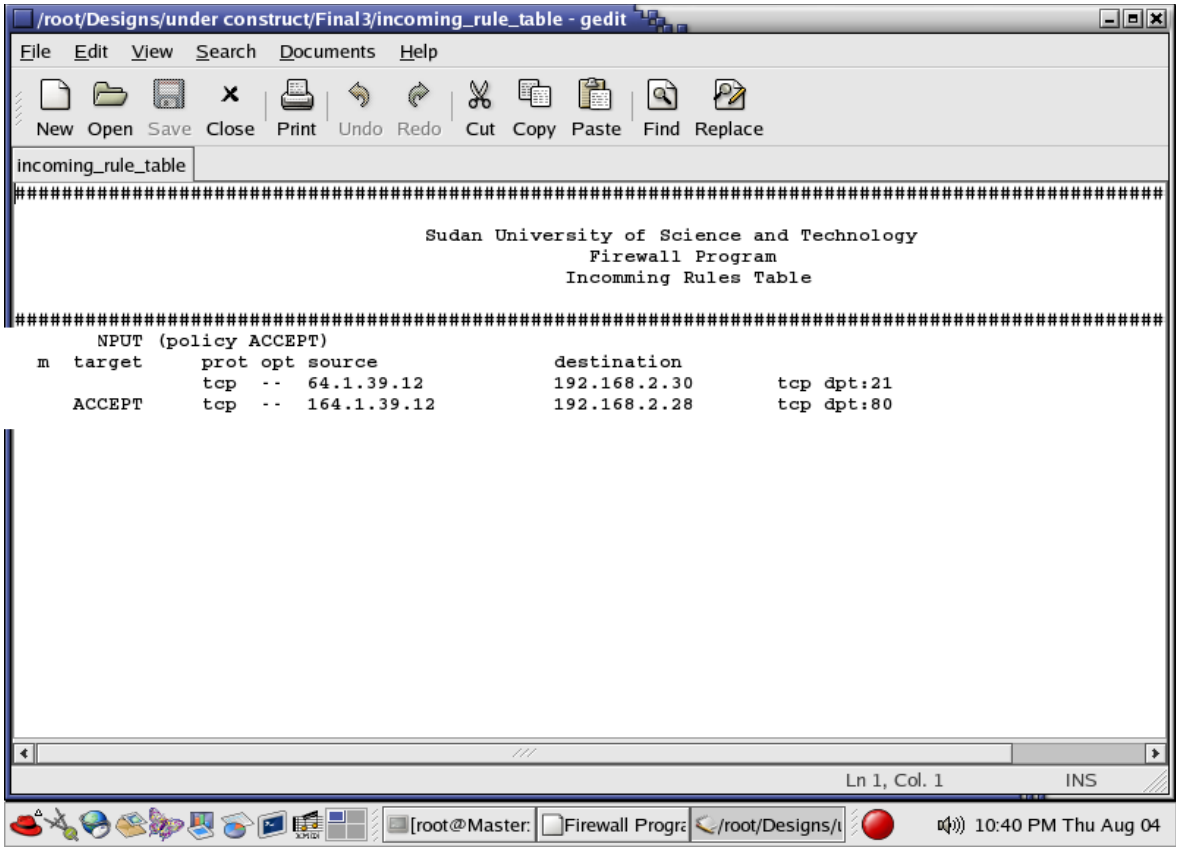


Figure E4 Graphical User Interface for viewing Incoming Rules

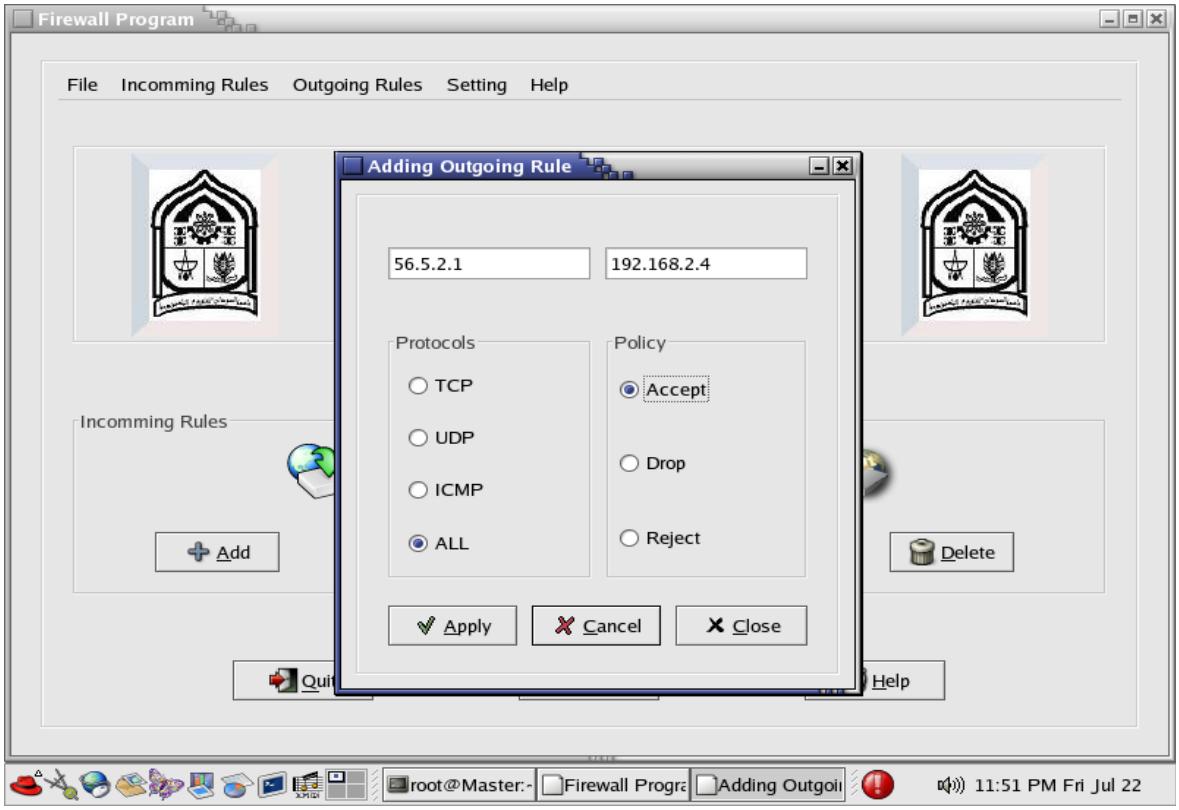


Figure E5 Graphical User Interface for Adding Outgoing Rules.

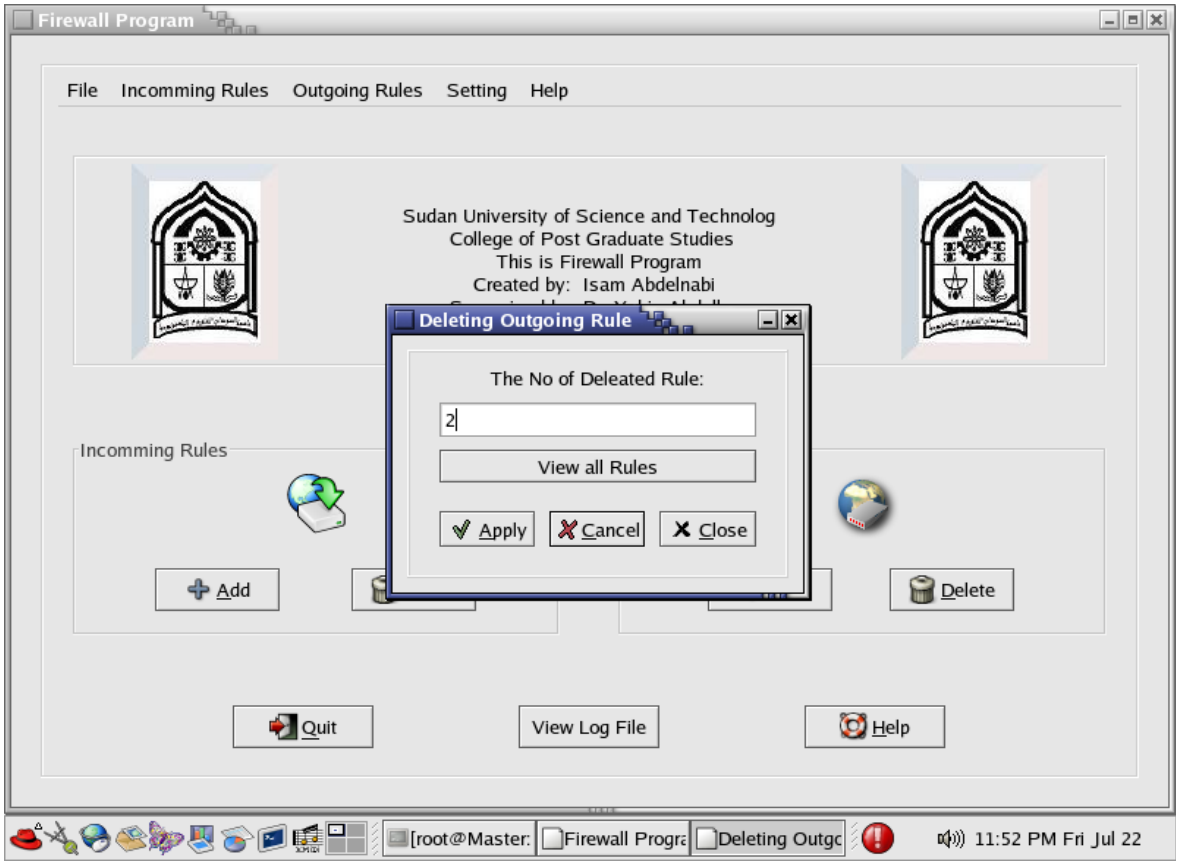


Figure E6 Graphical User Interface for Deleting Outgoing Rules.

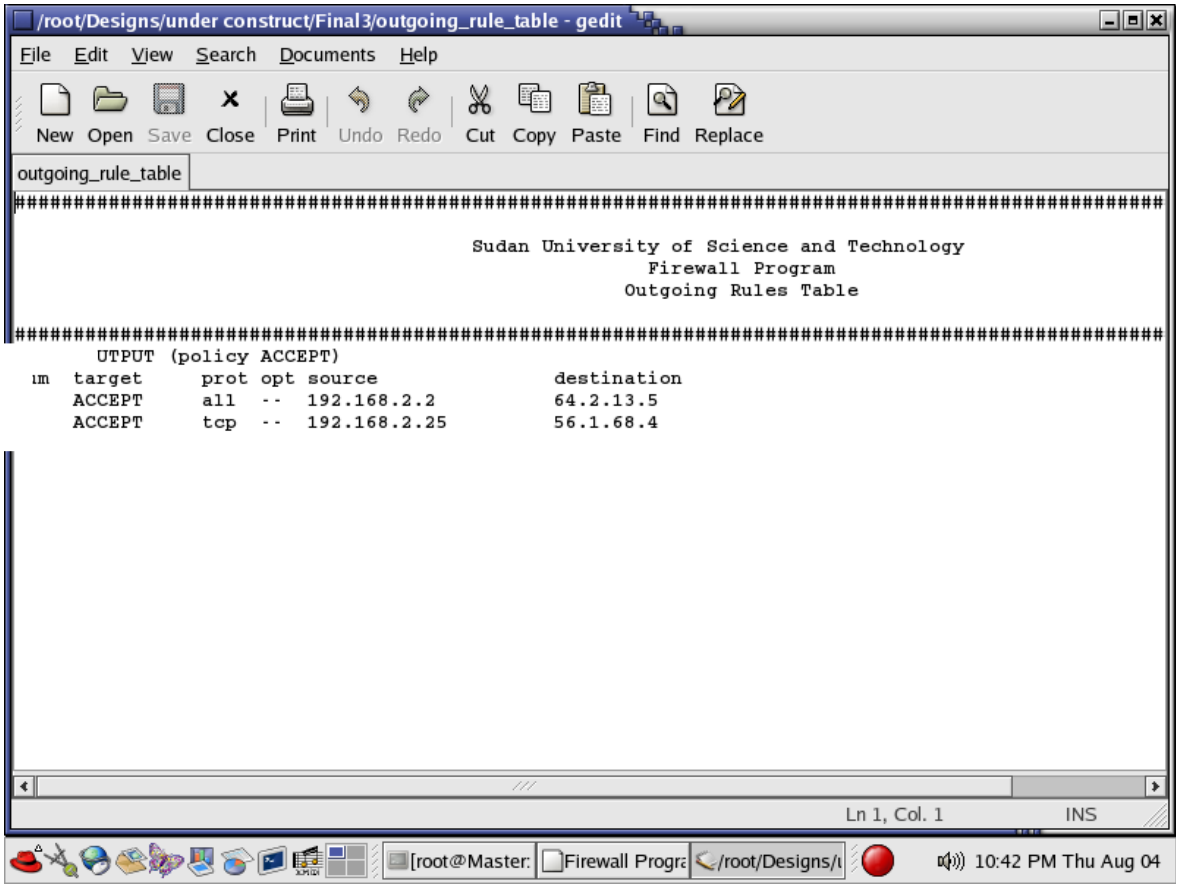


Figure E7 Graphical User Interface for viewing Outgoing Rules

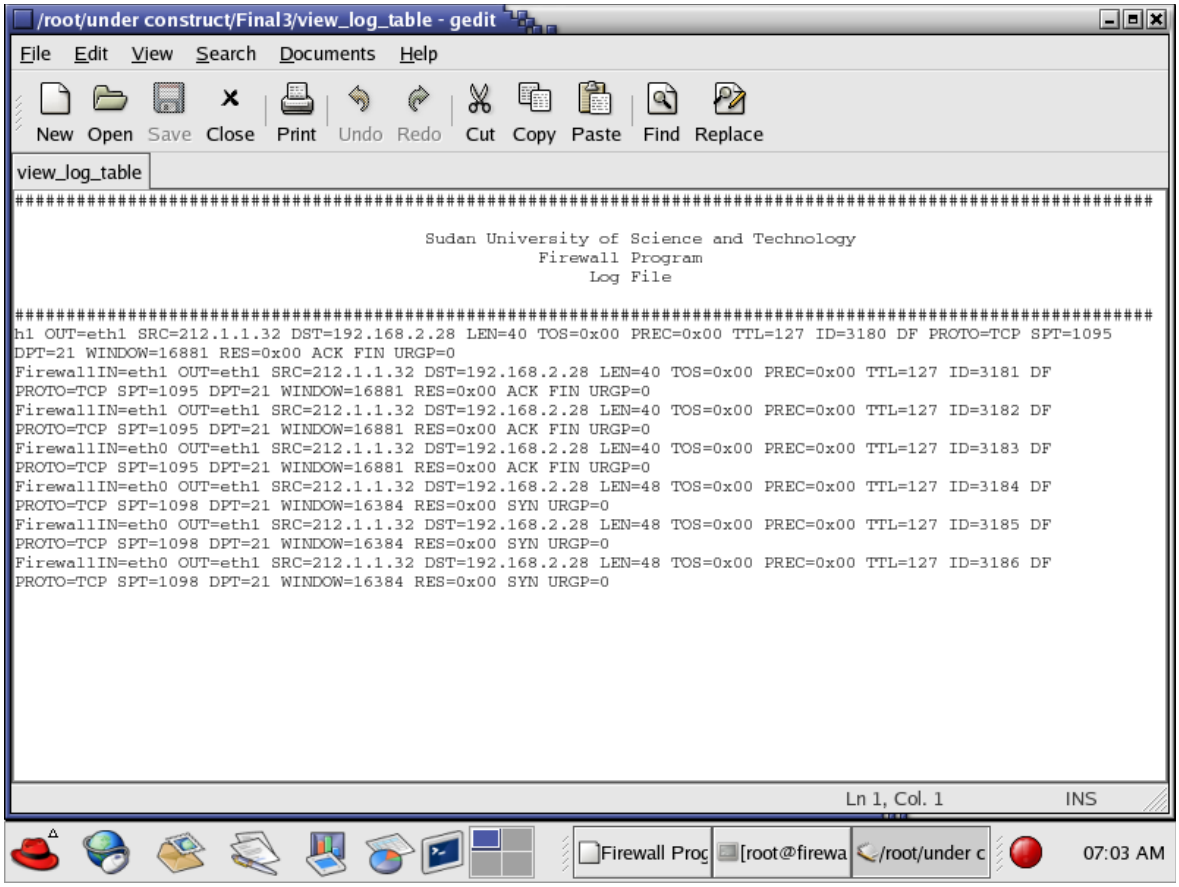


Figure E8 Graphical User Interface for viewing Logging File.

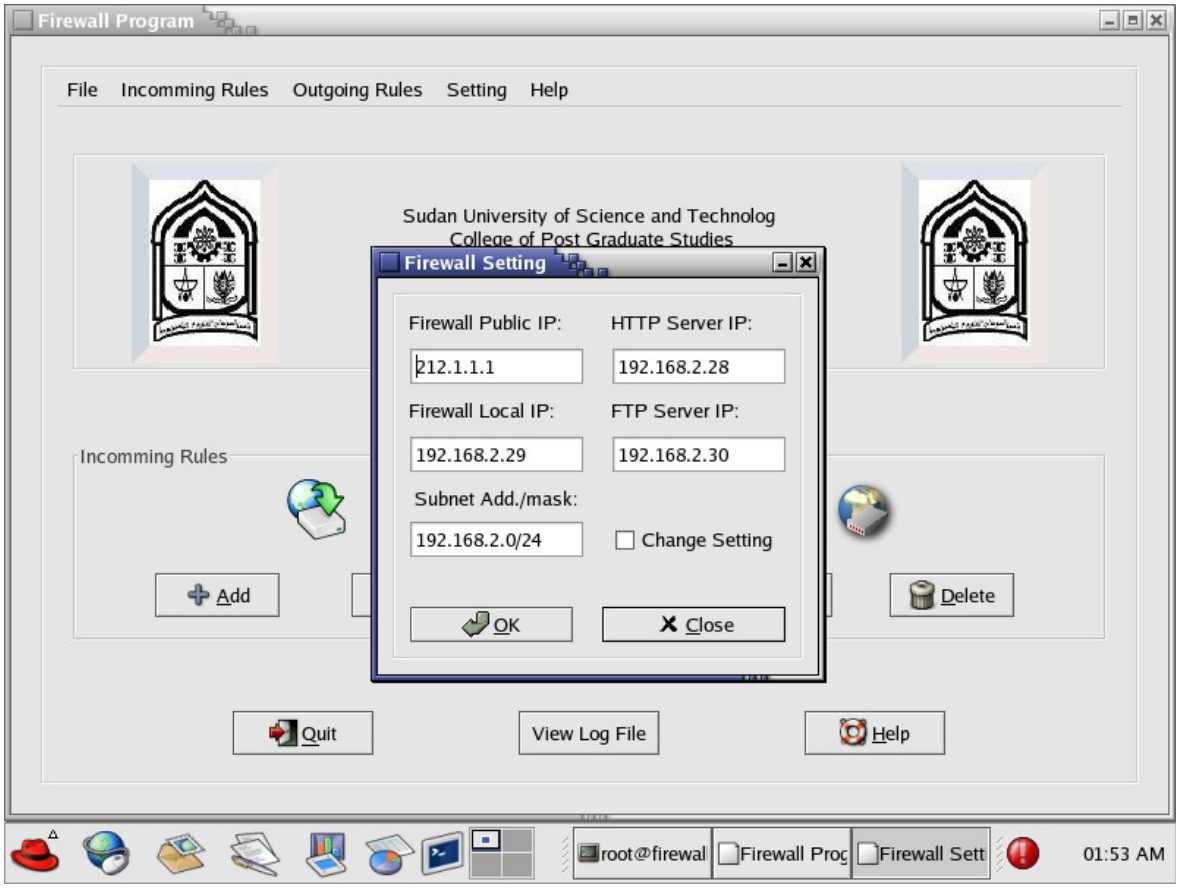


Figure E9 Graphical User Interface for Setting the Firewall.

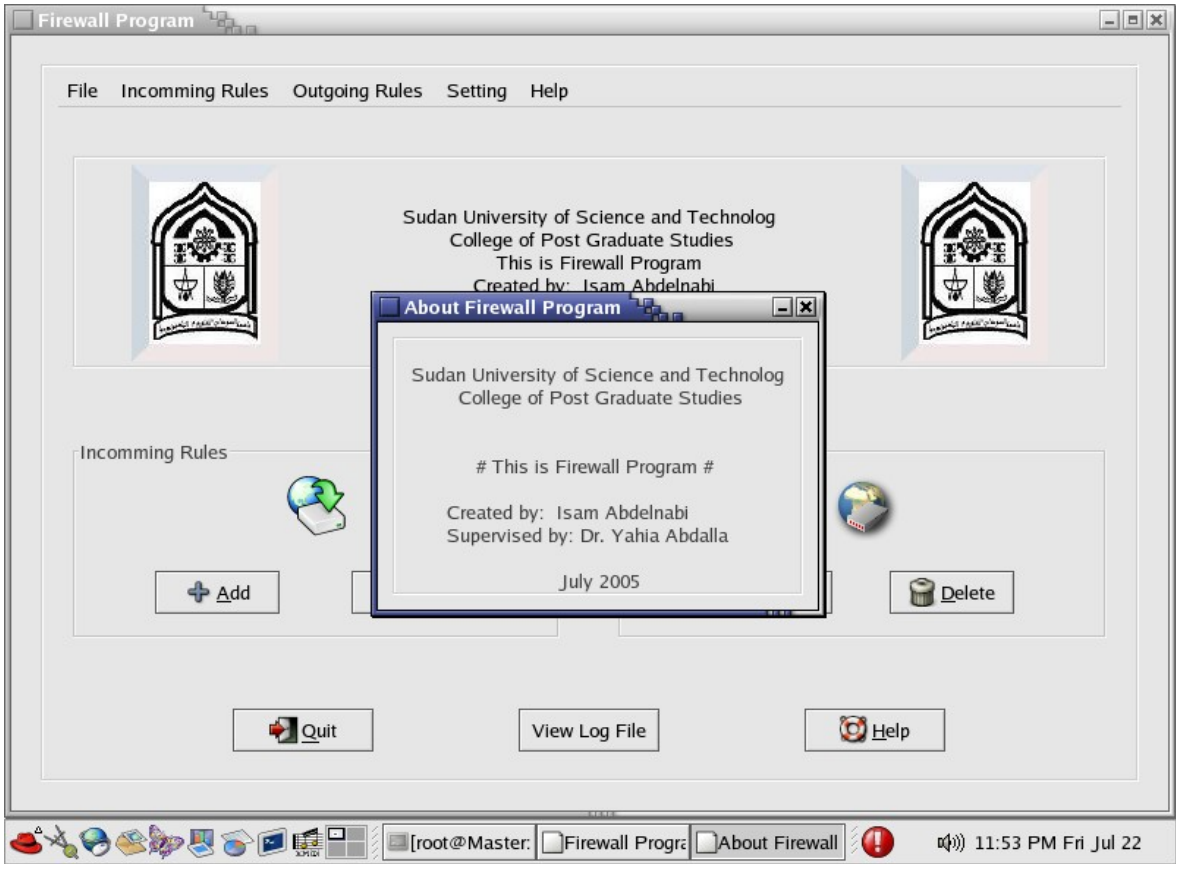


Figure E10 Graphical User Interface about the Firewall.