FIREWALL MANAGER

A PROJECT REPORT

Submitted by

Govindaraj.V	71201205017
Shiyam Kumar.K.N	71201205055
Somasundaram.S	71201205058

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

Kumaraguru College of Technology, Coimbatore

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2005

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"FIREWALL MANAGER"** is the bonafide work of **"GOVINDARAJ.V , SHIYAM KUMAR. K.N, SOMASUNDARAM.S"** who carried out the project work under my supervision.

[illegible]	[illegible] 15/4/05
SIGNATURE	**SIGNATURE**
Prof. Dr. S.Thangasamy	Ms.U.Sakthi
HEAD OF THE DEPARTMENT	**SUPERVISOR**
	Lecturer
Computer Science and Engineering	Information Technology
Kumaraguru College of Technology	Kumaraguru College of Technology
Coimbatore - 641006.	Coimbatore - 641006.

The candidates with University Register Nos. **71201205017, 71201205055, 71201205058** were examined by us in the project viva-voce examination held on ...19.04.2005

[illegible]	[illegible]
INTERNAL EXAMINER	EXTERNAL EXAMINER
	19/04/05

DECLARATION

We,

Govindaraj.V	**71201205017**
Shiyam Kumar.K.N	**71201205055**
Somasundaram.S	**71201205058**

Declare that the project entitled "**FIREWALL MANAGER**", submitted in partial fulfillment to Anna University as the project work of **Bachelor Of Technology (Information Technology) Degree**, is a record of original work done by us under the supervision and guidance of Ms.U.Sakthi,B.E., Senior lecturers, Department of Information Technology, Kumaraguru College Of Technology, Coimbatore.

Place : Coimbatore

Date : 15-04-2005

[illegible]

[Govindaraj. V]

[illegible]

[Shiyam Kumar. K. N]

[illegible]

[Somasundaram. S]

Project Guided by

[illegible] 15/4/05

[Ms. U. Sakthi]

ACKNOWLEDGEMENT

We are greatly indebted to our revered Principal **Dr.K.K.Padmanabhan, Ph.D.**, who has been the motivating force behind all our deeds.

We earnestly express our sincere thanks to our beloved Head of Department Prof. **Dr.S.Thangasamy, Ph.D** for his immense encouragement and help and for being our source of inspiration all through our course of study.

We are much obliged to express our sincere thanks and gratitude to our beloved guide **Ms.U.Sakthi., B.E.** for her valuable suggestions, constructive criticisms and encouragement which has enabled us to complete our project successfully.

We gratefully thank our Project Coordinator **Mr.K.R.Baskaran., M.S.** and our Class Advisor **Mrs.N.Chithradevi., M.E.** for extending their most appreciative and timely help to us.

We also thank all the staff members of the Department of Information Technology for all their encouragement and moral support.

We also extend our heartiest thanks to all our friends for their continuous help and encouragement throughout the course of study.

ABSTRACT

Firewall Manager is a desktop application that allows a user to create and manage linux- based firewalls. Firewall Manager is designed to handle environments with a single firewall and few rules, all the way to a large environment with many firewalls and complex rulebases. Either one should be able to be deployed and managed easily and securely, with a minimum of effort. Firewall Manager represents all rulebases (or policies as we'll call them) as one or more simple tables, populated with graphical objects representing various objects on your network. For example a web server would be defined as a Host, and there are also Firewall, Network, and Group objects available. Firewall Manager can then generate the scripts necessary to configure iptables on your remote firewalls, and securely transfer those scripts and execute them on the firewall. This application allows a user to monitor multiple firewalls easily from one client console

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OFABBREVIATIONS

1	IP	Internet Protocol
2	TCP	Transmission Control Protocol
3	UDP	User Datagram Protocol
4	ICMP	Internet Control Messaging Protocol
5	HTTP	Hyper Text Transfer Protocol
6	FTP	File Transfer Protocol
7	SSH	Secure Shell
8	SSHD	Secure Shell Deamon
9	NAT	Network Address Translation
10	SNAT	Source Network Address Translation
11	DNAT	Destination Network Address Translation

1. INTRODUCTION

1.1. Existing system and its Limitations

Network security is a primary consideration in any decision to host a website as the threats are becoming more widespread and persistent every day. One means of providing additional protection is to invest in a firewall.

The default firewall in Linux is iptables. iptables is a generic table structure for the definition of rulesets. Each rule within an IP table consists of a number of classifiers (iptables matches) and one connected action (iptables target).

Tables available in iptables:

All packets inspected by iptables pass through a sequence of built-in tables (queues) for processing. Each of these queues is dedicated to a particular type of packet activity and is controlled by an associated packet transformation/filtering chain.

There are three tables in total. The first is the mangle table which is responsible for the alteration of quality of service bits in the TCP header. This is hardly used in a home or SOHO environment.

The second table is the filter queue which is responsible for packet filtering. It has three built-in chains in which you can place your firewall policy rules. These are :

- Forward chain
- Input chain
- Output chain

The third table is the nat queue which is responsible for network address translation. It has two built-in chains, these are:

- Pre-routing chain
- Post-routing chain

Limitations

1) iptables is command driven.
2) Difficult to configure.
3) More overhead for Administrators.
4) Less flexible and scalable.
5) Difficult to manage.
6) Time consuming process.
7) Module dependency.

1.2. Proposed system and its advantages

The proposed system represents all rulebases as one or more simple tables, populated with graphical objects representing various objects on your network..

Features

- Create objects in object tree
- Create a Managed Firewall object
- Add and/or edit firewall rules
- Add and/or edit NAT rules
- Generate Scripts
- Install and Uninstall Policy

Advantages

- It is easy to maintain and configure
- Easy to install and uninstall the scripts
- It provides less overhead for administrators
- More user friendliness and less time consuming
- Easily understandable by all novice users

2. SOFTWARE REQUIREMENTS SPECIFICATION

2.1 Introduction

2.1.1 Purpose

The purpose of this document is to specify the requirements of project "Firewall Manager". It describes the interfaces for the system. The document also bridges the communication gap between the customer and the analyst.

2.1.2 Scope

SRS forms the basis for agreement between the client and the supplier and what the software product will do. It also provides a reference for the validation of the final project.

Any changes made to the SRS in the future will have to go through formal change approval process.

2.1.3 Definition

Customer:

A person or organization, internal or external to the producing organization, who takes financial responsibility for the system. In a large system this may not be the end user. The customer is the ultimate recipient of the developed product and its artifacts.

User :

A person who will use the system that is developed.

Analyst :

The Analyst details the specification of the system's functionality by describing the requirements aspect and other supporting software requirements.

2.1.4 Abbreviation

SRS	:	Software Requirement Specification
JDK	:	Java Development Kit
SSHD	:	Secure Shell Daemon
LAN	:	Local Area Network
GUI	:	Graphical User Interface

2.1.5 References

"An integrated approach to software engineering" by Pankaj Jalote.

"Software Engineering" by Roger S. Pressman.

2.2 General Description

2.2.1 Product Overview

This system aims to provide the administrator complete control of his LAN through the server. It is a GUI based client firewall particularly in large, complex environment. Security policies are easy to build, view and audit, as well as common tasks normally done on the remote firewall.

2.2.2 User Characteristics

The main user of this system is the administrator who is in charge of the network. He is expected to have a basic knowledge of firewall and its operations.

2.2.3 General Constraints

Initially this is designed and tested on *nix-based clients. Windows support for remote firewall management is not on the short list of objectives.

2.3 Specific Requirements

2.3.1 Inputs and Outputs

The input to the system are the host objects, services, network and firewall. Administrator privilege is used for installing and running the scripts successfully at all run levels.

The output from the system are the iptables compatible scripts to be executed at the run level.

2.3.2 Functional requirements:

i. The system should be able to authenticate the network administrator and control the network.

ii. The system should allow the network administrator to perform the desired functions with ease.

2.3.3 Performance constraints:

The modules such as CONFIG_IP_NF_CONNTRACK, CONFIG_IP_NF_FTP, CONFIG_IP_NF_IPTABLES, CONFIG_IP_NF_FILTER are required for the system to work efficiently.

2.3.4 Software Constraints:

The system runs on Unix based platforms and requires iptables and sshd running.

3 LITERATURE REVIEW

3.1 Firewall

A firewall is simply a host whose main purpose is to protect your network. A firewall restricts certain types of network traffic from the Internet to your protected network(s) - the reverse is also often true.

3.1.1 Types Of Exploits

- Local - There is no security without physical security. If someone has physical access to your box, you've lost. Obviously, a firewall won't help you here.
- Local privilege escalation - The trojan horse attack. The attacker already has a local account on your box (inside the gates) and obtains root by some means (vulnerability or misconfiguration). A firewall cannot protect again this type of attacks.
- Remote - Your host is listening on a port that the attacker is able to connect to remotely over a network and exploit a vulnerability somehow. This is the only type of attack a firewall can (hopefully) protect you against. There is another important point here that most firewall often neglect. In order for someone to exploit your box remotely, it has to be listening on some ports (i.e. providing a way for an attacker to connect). Therefore, if your host isn't listening on any ports, you are safe from remote exploits (unless the attacker manages to attack the network stack itself).

3.1.2 Need For A Firewall

- Increase your network security - Some services are inherently insecure and impossible to secure on individual hosts. A firewall can help you segment and contain parts of your network to increase security.

- Network access control - A firewall can help you enforce your network security policies by selectively allowing network services (to all or selected hosts).
- Logging - Because a firewall must examine all inbound/outbound network traffic, it can help you log network activity (that passes through the firewall).

3.1.3 Types Of Firewalls

- Proxying firewall - Proxy servers work by making requests on behalf of your clients.
- Packet filtering firewall - Packet filters work by examining the IP packets .

3.2 Packet Filtering

A packet filter is a piece of software which looks at the packets as they pass through, and decides the fate of the entire packet. It might decide to DROP the packet (i.e., discard the packet as if it had never received it), ACCEPT the packet (i.e., let the packet go through), or something more complicated.

3.2.1 Advantages Of Packet Filtering

CONTROL:

When you are using a Linux box to connect your internal network to another network (say, the Internet) you have an opportunity to allow certain types of traffic, and disallow others. For example, the header of a packet contains the destination Address of the packet, so you can prevent packets going to a certain part of the outside network.

SECURITY:

When your Linux box is the only thing between the chaos of the Internet and your nice, orderly network, it's nice to know you can restrict what comes tromping in your door. For example, you might allow anything to go out from your network, but you might be worried about the well-known `Ping of Death' coming in from malicious outsiders.

As another example, you might not want outsiders telnetting to your Linux box, even though all your accounts have passwords. Maybe you want (like most people) to be an observer on the Internet, and not a server (willing or otherwise). Simply don't let anyone connect in, by having the packet filter reject incoming packets used to set up connections.

WATCHFULNESS:

Sometimes a badly configured machine on the local network will decide to spew packets to the outside world. It's nice to tell the packet filter to let you know if anything abnormal occurs; maybe you can do something about it, or maybe you're just curious by nature.

4. FIREWALL MANAGER

4.1 Architecture

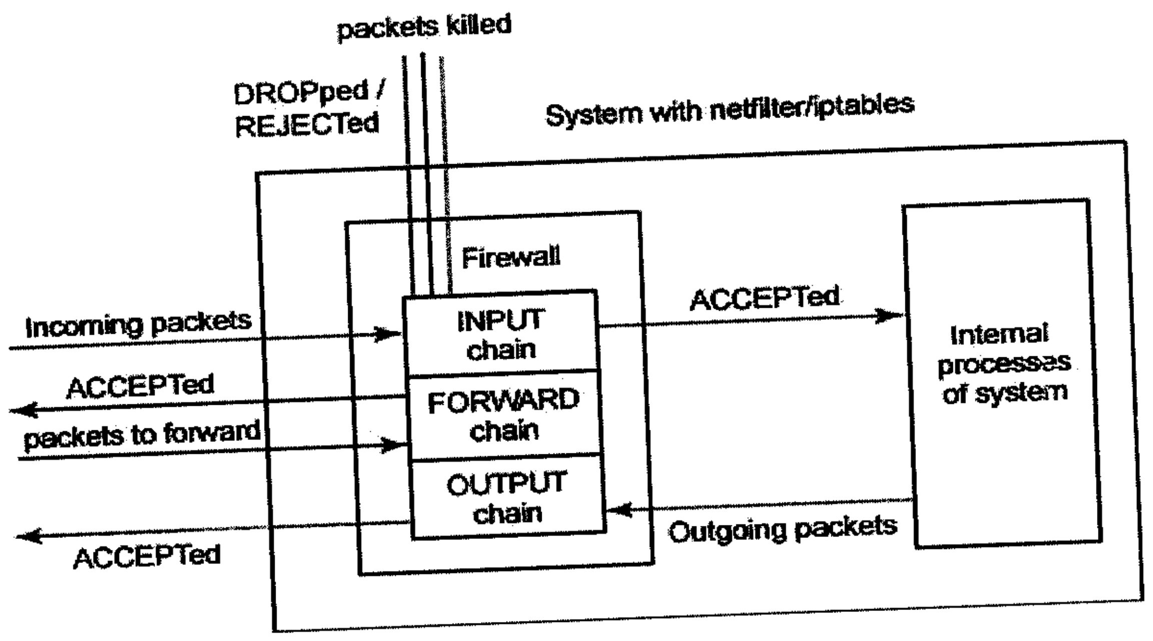


Fig No.4.1 Architecture of Firewall Manager

4.2 Packet Processing

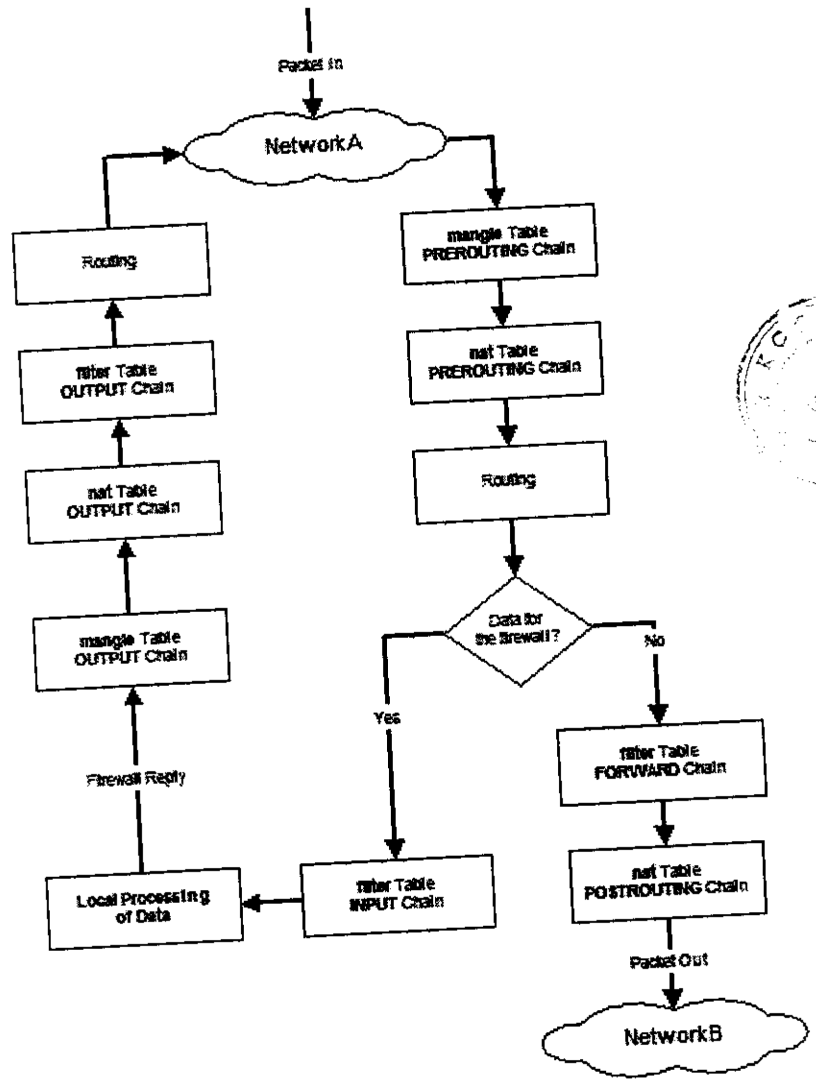


Fig No.4.2 Flowchart for Packet processing

The packet is first examined by your rules in the mangle table's PREROUTING chain, if any. It is then inspected by the rules in the nat table's PREROUTING chain to see whether the packet requires DNAT. It is then routed.

If the packet is destined for a protected network, then it is filtered by the rules in the FORWARD chain of the filter table and, if necessary, the packet undergoes SNAT before arriving at Network B. When the destination server decides to reply, the packet undergoes the same sequence of steps.

If the packet is destined for the firewall itself, then it is filtered by the rules in the INPUT chain of the filter table before being processed by the intended application on the firewall. At some point, the firewall needs to reply. This reply is inspected by your rules in the OUTPUT chain of the mangle table, if any. The rules in the OUTPUT chain of the nat table determine whether address translation is required and the rules in the OUTPUT chain of the filter table are then inspected before the packet is routed back to the Internet.

44.3 Packet Filtering Tables

Queue Type	Queue Function	Packet Transformation chain in Queue	Chain Function
Filter	Packet filtering	FORWARD	Filters packets to servers accessible by another NIC on the firewall.
		INPUT	Filters packets destined to the firewall.
		OUTPUT	Filters packets originating from the firewall
Nat	Network Address Translation	PREROUTING	Address translation occurs before routing. Facilitates the transformation of the destination IP address to be compatible with the firewall's routing table. Used with NAT of the destination IP address, also known as destination NAT or DNAT.
		POSTROUTING	Address translation occurs after routing. This implies

				that there was no need to modify the destination IP address of the packet as in pre-routing. Used with NAT of the source IP address using either one-to-one or many-to-one NAT. This is known as source NAT, or SNAT.
			OUTPUT	Network address translation for packets generated by the firewall.

4.4. Displacement of rules to different chains

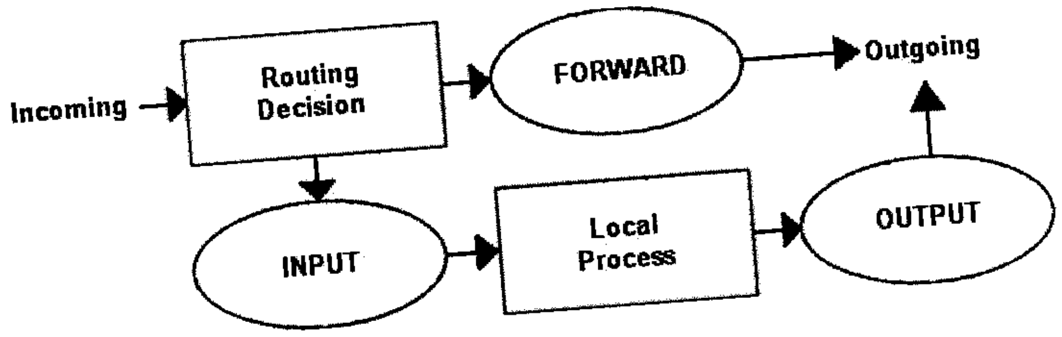


Fig No.4.3 General Packet Processing

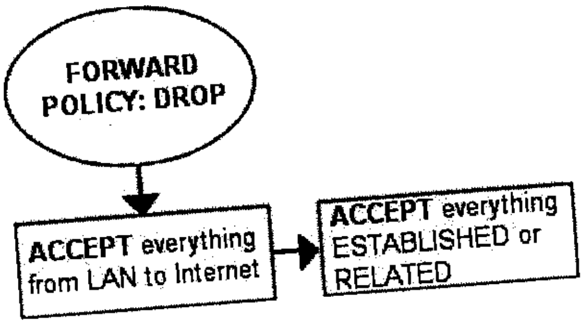

FORWARD POLICY: DROP
ACCEPT everything from LAN to Internet
ACCEPT everything ESTABLISHED or RELATED

Fig No.4.4 Forward chain

We want the local network to be able to connect to the Internet, of course. To do this, we will need to NAT all packets since none of the local computers have real IP addresses. All of this is done within the PREROUTING chain. . This means that we will also have to do some filtering within the FORWARD chain since we will otherwise allow outsiders full access to our local network. We trust our local network to the fullest, and because of that we specifically allow all traffic from our local network to the Internet. Since no one on the Internet should be allowed to contact our local network computers, we will want to block all traffic from the Internet to our local network except already established and related connections, which in turn will allow all return traffic from the Internet to our local network.

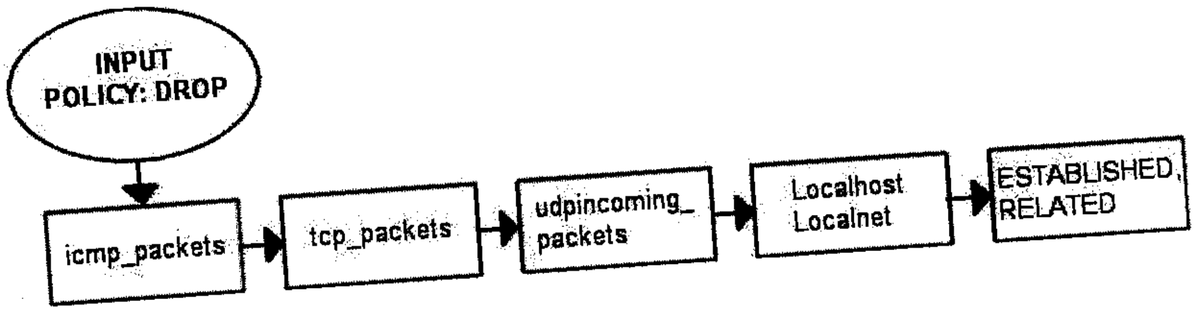


Fig No.4.5 Input chain

As for our firewall, we just want to offer a few services to people on the Internet. Therefore, we have decided to allow HTTP, FTP, SSH access to the actual firewall. All of these protocols are available on the actual firewall, and hence it should be allowed through the INPUT chain, and we need to allow the return traffic through the OUTPUT chain. However, we also trust the local network fully, and the loopback device and IP address are also trusted. Because of this, we want to add special rules to allow all traffic from the local network as well as the loopback network interface. Also, we do not want to allow specific packets or packet headers in specific conjunctions, nor do we want to allow some IP ranges to reach the firewall from the Internet. For instance, the 10.0.0.0/8 address range is reserved for local networks and hence we would normally not want to allow packets from such a address range since they would with 90% certainty be spoofed.

Since we have an FTP server running on the server, as well as the fact we want to traverse as few rules as possible, we add a rule which lets all established and related traffic through at the top of the INPUT chain. For the same reason, we want to split the rules down into sub-chains. By doing this, our packets will hopefully only need to traverse as few rules as possible. By traversing less rules, we make the rule-set less time consuming for each packet, and reduce redundancy within the network.

We chose to split the different packets down by their protocol family, for example TCP, UDP or ICMP. All TCP packets traverse a specific chain named tcp_packets, which will contain rules for all TCP ports and protocols that we want to allow. Also, we want to do some extra checking on the TCP packets, so we would like to create one more subchain for all packets that are accepted for using valid port numbers to the firewall. This chain we choose to call the allowed chain, and should contain a few extra checks before finally accepting the packet. As for ICMP packets, these will traverse the icmp_packets chain. When we decided on how to create this chain, we could not see any specific needs for extra checks before allowing the ICMP packets through if we agree with the type and code of the ICMP packet, and hence we accept them directly. Finally, we have the UDP packets which need to be dealt with. These packets, we send to the udp_packets chain which handles all incoming UDP packets. All incoming UDP packets should be sent to this chain, and if they are of an allowed type we should accept them immediately without any further checking.

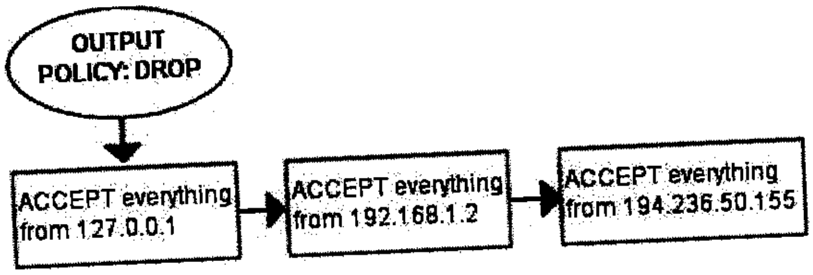

OUTPUT
POLICY: DROP
ACCEPT everything from 127.0.0.1
ACCEPT everything from 192.168.1.2
ACCEPT everything from 194.236.50.155

Fig No.4.6 Output chain

Finally, we have the firewalls OUTPUT chain. Since we actually trust the firewall quite a lot, we allow pretty much all traffic leaving the firewall. We do not do any specific user blocking, nor do we do any blocking of specific protocols. However, we do not want people to use this box to spoof packets leaving the firewall itself, and hence we only want to allow traffic from the IP addresses assigned to the firewall itself. We would most likely implement this by adding rules that ACCEPT all packets leaving the firewall in case they come from one of the IP addresses assigned to the firewall, and if not they will be dropped by the default policy in the OUTPUT chain.

4.5. Using Firewall Manager

4.5.1 Overview

a) Create objects in object tree

b) Create a Managed Firewall object

c) Add and/or edit firewall rules

d) Add and/or edit NAT rules

e) Install Policy

4.5.2 Creating Objects

There are a few major categories of objects you can create, some with sub-categories. Network Objects represent networked objects with IP addresses. The sub-classes include Host (single IP), Network (contiguous range of IPs), Managed Firewall , and Group (any combination of the above objects). Service objects represent a certain type of IP connection, in the case of TCP or UDP that includes source and destination ports, and for ICMP that includes ICMP type and code.To create an object, right-click on the folder in the tree in the left panel of Firewall Manager and click ADD. Define your object in the resulting dialog box and click OK.Common services come pre-defined, but you will need to define hosts, networks, and at least one Managed Firewall (required to generate scripts and install policies).

4.5.3 Creating Firewall Rules

Click Edit -> Add Rule, to add a new blank rule to your current active policy. Some cells can be edited by right-clicking the cell and choosing Edit Cell. Rates and Actions are just drop down boxes, Log is just a checkbox, and comments can be entered by clicking on the comment cell and typing. The firewall will look for packets matching the source, destination, service and rate of the packet. If it gets a match, it will do the action chosen in the Action column. Accept means the packet will be forwarded as necessary, Drop means the packet will not be allowed to pass, and Deny means the packet will not be allowed to pass, but a message will be sent back to the source, notifying it that the packet was not allowed. In addition, the match will also be logged if the logged box is checked. If the packet is not matched, the firewall will consult the next rule down the list. By default, at the end of every rule list, Firewall Manager configures the firewall to drop anything that doesn't match. Firewall Manager tries to account for statefullness. Meaning that responses from a TCP connection are automatically allowed to return, without you needing to specify a rule for that. For UDP, a rule is opened for return packets also, even though UDP is stateless. For ICMP, rules need to be created in both directions.

4.5.4 Creating NAT Rules

Creating NAT rules are similar to firewall rules, but click on the NAT tab in the main window first. In NAT rules though, the firewall will look for matches in the Original columns (Source, Destination and Service), and if it gets a match, will change the IPs to the corresponding Translated column entry.

4.5.5 Saving Policy

Use File -> Save or File -> Save As to save your policy. This creates a local copy of your policy for later use. This should be done before trying to install your policy.

4.5.6 Installing Policy

Use Policy -> Install to install your policy. A list of all your defined ManagedFirewalls will be presented, and you can choose one or more to install to. This will create a local copy of your iptables script, copy it to the remote firewall (using scp or sftp, so sshd will need to be running on the remote firewall), chmod it if required and execute it to make it active immediately. Alternatively, you can use Policy -> Generate Scripts to create a local copy of your iptables scripts, and then copy and execute manually.

5. FEATURES OF FIREWALL MANAGER

Firewall Manager is easily customizable. This means that you can add or remove filters based on several conditions. Some of these are:

- IP ADDRESS - Each machine on the Internet is assigned a unique address called an **IP address**. IP addresses are 32-bit numbers, normally expressed as four "octets" in a "dotted decimal number." A typical IP address looks like this: 216.27.61.137. For example, if a certain IP address outside the company is reading too many files from a server, the firewall can block all traffic to or from that IP address.
- DOMAIN NAMES - Because it is hard to remember the string of numbers that make up an IP address, and because IP addresses sometimes need to change, all servers on the Internet also have human-readable names, called **domain names**. For example, it is easier for most of us to remember www.linwin2k.4t.com than it is to remember 216.27.61.137. A company might block all access to certain domain names, or allow access only to specific domain names.
- PROTOCOLS - The **protocol** is the pre-defined way that someone who wants to use a service talks with that service. The "someone" could be a person, but more often it is a computer program like a Web browser. Protocols are often text, and simply describe how the client and server will have their conversation. The **http** in the Web's protocol. Some common protocols that you can set firewall filters for include:

- **IP** (Internet Protocol) - the main delivery system for information over the Internet
- **TCP** (Transmission Control Protocol) - used to break apart and rebuild information that travels over the Internet
- **HTTP** (Hyper Text Transfer Protocol) - used for Web pages
- **FTP** (File Transfer Protocol) - used to download and upload files
- **UDP** (User Datagram Protocol) - used for information that requires no response, such as streaming audio and video
- **ICMP** (Internet Control Message Protocol) - used by a router to exchange the information with other routers
- **SMTP** (Simple Mail Transport Protocol) - used to send text-based information (e-mail)
- **SNMP** (Simple Network Management Protocol) - used to collect system information from a remote computer
- **Telnet** - used to perform commands on a remote computer

A company might set up only one or two machines to handle a specific protocol and ban that protocol on all other machines.

- PORTS - Any server machine makes its services available to the Internet using numbered **ports**, one for each service that is available on the server (see How_Web Servers Work for details). For example, if a server machine is running a Web (HTTP) server and an FTP server, the Web server would typically be available on port 80, and the FTP server would be available on port 21. A company might block port 21 access on all machines but one inside the company.

Firewall Manager is a software firewall that can be installed on the computer in your home that has an Internet connection. This computer is considered a gateway because it provides the only point of access between your home network and the Internet.

6. PRODUCT TESTING

Testing is done to detect the errors in the software. This implies not only to the coding phase but to uncover errors introduced in all the previous phases.

The following are the types of tests that were performed-

Unit testing: Each and every module is tested separately to check if its intended functionality is met. Some unit testing performed are,

- Checking the proper working of the iptables and sshd.
- Checking for proper connectivity between server and client .
- Ensuring that the input, output and forward module works and does not interfere with other functions.
- Checking the working of the services such as Telnet, FTP, SMTP, HTTP, PING.

Integration testing: It is the testing performed to detect errors on interconnection between modules. Here, the system is integrated with iptables and kernel, and tested that the scripts are executed in the runlevel.

System testing: The system is tested against the system requirements to see if all the requirements are met and if the system performs as per the specified requirements. The system is tested as a whole to check for its functionality.

Validation testing: This test is done to check for the validity of the entered input. The input to this system are the host IPAddress, Protocol type and the administrator's user id and password. Invalid characters and symbols are recognized and properly handled.

7. FUTURE ENHANCEMENTS

- Deploy and run scripts on remote firewalls
- Perform certain management or status tasks on remote firewalls
- Auditing of all policy changes
- Undo changes functionality
- Importing of objects, services, or rules from other policies
- Drag and Drop
- IPv6 support
- Support for other types of firewalls besides iptables.

8. CONCLUSION

A firewall is a critical part of any establishment that connects to an unprotected network such as the Internet. This application can help administrators create a strong firewall with the powerful, kernel-based firewall software. This application reduces the administrator's time in setting up a firewall thereby improving the security of the network. It is scalable and can be deployed in any environment ranging from home to larger networks. This enables administrators to protect their servers from a wide variety of hazards, including service attacks and hack attempts.

This application finds its usage in software industry where they want to protect their valuable data from both internal and external malicious users. It also enables particular services to be allowed or denied thus enriching more security to the network. This application also finds use in educational institutes where they want to restrict the user from viewing unethical sites. It can also be used in web browsing centers, home etc.

APPENDIX 1

Adding Rules

```
import javax.swing.JTabbedPane;
import javax.swing.AbstractAction;
import javax.swing.JFrame;
import java.awt.event.ActionEvent;
import javax.swing.event.ChangeListener;
import javax.swing.event.ChangeEvent;
import java.util.Vector;
import java.util.Enumeration;
import java.util.List;


public class TableAdd extends AbstractAction implements ChangeListener {

  private JWallMain jwall;
  private int mode = 0;
  public final static int MODE_FW     = 0;
  public final static int MODE_NAT    = 1;

  public TableAdd(JWallMain jwall) {
    super(PropertyManager.getString("menu.table.add.object"));
    this.jwall = jwall;
  }


  public void actionPerformed(final ActionEvent e) {
    Policy policy = PolicyManager.getInstance().getActivePolicy();
    switch (mode) {
      case MODE_FW:
        FWTable fwTable = jwall.getFWTable();
        int col = fwTable.getSelectedColumn();
        int row = fwTable.getSelectedRow();
        FWRule rule = fwTable.getFWRule(row);
        Vector to = new Vector();

        if (col == FWTable.COL_SRC) {
          Enumeration en = rule.enumSources();
        while (en.hasMoreElements()) {
          to.add(en.nextElement());
        }
        Vector from = new Vector();
        NetworkObject[] nos = policy.getNetworkObjects();
        for (int i = 0; i < nos.length; i++) {
          if (!to.contains(nos[i])) {
            from.add(nos[i]);
          }
        }
        MultiObjectChooser moc =
MultiObjectChooser.getInstance((JFrame) jwall,
              PropertyManager.getString("fwcol.1"),
              PropertyManager.getString("gui.moc.src.msg"),
              from, to, true);
        List list = moc.getSelection();
        if (list != null) {
          rule.setSources(list);
          fwTable.adjustRowHeights();
        }
        MultiObjectChooser.releaseInstance();

      } else if (col == FWTable.COL_DEST) {
        Enumeration en = rule.enumDestinations();
        while (en.hasMoreElements()) {
          to.add(en.nextElement());
        }
        Vector from = new Vector();
        NetworkObject[] nos = policy.getNetworkObjects();
        for (int i = 0; i < nos.length; i++) {
          if (!to.contains(nos[i])) {
            from.add(nos[i]);
          }
        }
        MultiObjectChooser moc =
MultiObjectChooser.getInstance((JFrame) jwall,
              PropertyManager.getString("fwcol.2"),
              PropertyManager.getString("gui.moc.dest.msg"),
              from, to, true);
        List list = moc.getSelection();
        if (list != null) {
          rule.setDestinations(list);
          fwTable.adjustRowHeights();
        }
        MultiObjectChooser.releaseInstance();
      } else if (col == FWTable.COL_SERV) {
        Enumeration en = rule.enumServices();
        while (en.hasMoreElements()) {
          to.add(en.nextElement());
        }
        Vector from = new Vector();
        Service[] servs = policy.getServices();
        for (int i = 0; i < servs.length; i++) {
          if (!to.contains(servs[i])) {
            from.add(servs[i]);
          }
        }
        MultiObjectChooser moc =
MultiObjectChooser.getInstance((JFrame) jwall,
                PropertyManager.getString("fwcol.3"),
                PropertyManager.getString("gui.moc.serv.msg"),
                from, to, true);
        List list = moc.getSelection();
        if (list != null) {
          rule.setServices(list);
          fwTable.adjustRowHeights();
        }
        MultiObjectChooser.releaseInstance();
      }
      break;
    case MODE_NAT:
      NATTable natTable = jwall.getNATTable();
      int natCol = natTable.getSelectedColumn();
      int natRow = natTable.getSelectedRow();
      NATRule natRule = natTable.getNATRule(natRow);
      Vector natTo = new Vector();
      if (natCol == NATTable.COL_O_SRC) {
        Enumeration en = natRule.enumOriginalSources();
        while (en.hasMoreElements()) {
          natTo.add(en.nextElement());
```

```
            }
            Vector from = new Vector();
            NetworkObject[] nos = policy.getNetworkObjects();
            for (int i = 0; i < nos.length; i++) {
                if (!natTo.contains(nos[i])) {
                    from.add(nos[i]);
                }
            }
            MultiObjectChooser moc =
MultiObjectChooser.getInstance(jwall,
                    PropertyManager.getString("natcol.1"),
                    PropertyManager.getString("gui.moc.src.msg"),
                    from, natTo, true);
            List list = moc.getSelection();
            if (list != null) {
                natRule.setSources(list);
                natTable.adjustRowHeights();
            }
            MultiObjectChooser.releaseInstance();

        } else if (natCol == NATTable.COL_O_DEST) {
            Enumeration en = natRule.enumOriginalDestinations();
            while (en.hasMoreElements()) {
                natTo.add(en.nextElement());
            }
            Vector from = new Vector();
            NetworkObject[] nos = policy.getNetworkObjects();
            for (int i = 0; i < nos.length; i++) {
                if (!natTo.contains(nos[i])) {
                    from.add(nos[i]);
                }
            }
            MultiObjectChooser moc =
MultiObjectChooser.getInstance((JFrame) jwall,
                    PropertyManager.getString("fwcol.2"),
                    PropertyManager.getString("gui.moc.dest.msg"),
                    from, natTo, true);
            List list = moc.getSelection();
            if (list != null) {
                natRule.setDestinations(list);
            natTable.adjustRowHeights();
          }
          MultiObjectChooser.releaseInstance();
        } else if (natCol == NATTable.COL_O_SERV) {
          Enumeration en = natRule.enumOriginalServices();
          while (en.hasMoreElements()) {
            natTo.add(en.nextElement());
          }
          Vector from = new Vector();
          Service[] servs = policy.getServices();
          for (int i = 0; i < servs.length; i++) {
            if (!natTo.contains(servs[i])) {
              from.add(servs[i]);
            }
          }
          MultiObjectChooser moc =
MultiObjectChooser.getInstance((JFrame) jwall,
                    PropertyManager.getString("fwcol.3"),
                    PropertyManager.getString("gui.moc.serv.msg"),
                    from, natTo, true);
            List list = moc.getSelection();
            if (list != null) {
              natRule.setServices(list);
              natTable.adjustRowHeights();
            }
            MultiObjectChooser.releaseInstance();
          } else if (natCol == NATTable.COL_X_SRC) {
            ObjectListDialog old = new ObjectListDialog(jwall,
 policy.getNetworkObjects(),
 PropertyManager.getString("gui.policy.choose"), false);
            Object[] obj = old.getSelectedObjects();
            if (obj.length > 0) {
              NetworkObject xSrc = (NetworkObject) obj[0];
              if (xSrc != null) {
                natRule.setXlatedSource(xSrc);
                natTable.adjustRowHeights();
                natTable.setValueAt(xSrc, natRow, natCol);
              }
            }
          } else if (natCol == NATTable.COL_X_DEST) {
                ObjectListDialog old = new ObjectListDialog(jwall,
policy.getNetworkObjects(),
PropertyManager.getString("gui.policy.choose"), false);
                Object[] obj = old.getSelectedObjects();
                if (obj.length > 0) {
                    NetworkObject xDest = (NetworkObject) obj[0];
                    if (xDest != null) {
                        natRule.setXlatedDestination(xDest);
                        natTable.adjustRowHeights();
                        natTable.setValueAt(xDest, natRow, natCol);
                    }
                }
            } else if (natCol == NATTable.COL_X_SERV) {
                ObjectListDialog old = new ObjectListDialog(jwall,
policy.getServices(), PropertyManager.getString("gui.policy.choose"),
false);
                Object[] obj = old.getSelectedObjects();
                if (obj.length > 0) {
                    Service xServ = (Service) obj[0];
                    if (xServ != null) {
                        natRule.setXlatedService(xServ);
                        natTable.adjustRowHeights();
                        natTable.setValueAt(xServ, natRow, natCol);
                    }
                }
            }
            break;
        }
    }


    public void stateChanged(ChangeEvent e) {
        JTabbedPane tab = (JTabbedPane) e.getSource();
        switch (tab.getSelectedIndex()) {
            case MODE_FW:
                mode = MODE_FW;
                break;
            case MODE_NAT:
                mode = MODE_NAT;
                break;
```

```
      }
    }
}
```

Policy Generate

```
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.util.logging.Level;
import java.util.logging.Logger;


public class PolicyGenerate extends AbstractAction {

    private static Logger logger = Logger.getLogger("org.jwall.gui.handler");
    private JFrame frame;
    private Preference prefs =
PreferenceManagerInstance.getInstance().getPreference();


    public PolicyGenerate(JFrame frame) {
        super(PropertyManager.getString("menu.policy.generate"));
        this.frame = frame;
    }

    public void actionPerformed(final ActionEvent e) {
        Policy activePolicy = PolicyManager.getInstance().getActivePolicy();
        ManagedFirewall[] firewalls = activePolicy.getFirewalls();
        Object[] list = new Object[firewalls.length];
        for (int i = 0; i < firewalls.length; i++) {
            list[i] = firewalls[i];
        }
            ObjectListDialog    old    =    new    ObjectListDialog(frame,    list,
PropertyManager.getString("gui.policy.choose"), true);
        Object[] selected = old.getSelectedObjects();
        for (int f = 0; f < selected.length; f++) {
        File fwScript = null;
        JFileChooser fc = new JFileChooser();
        fc.setCurrentDirectory(prefs.getDirConfiguration());
            int retVal = fc.showSaveDialog(frame);
        if (retVal == JFileChooser.APPROVE_OPTION) {
          fwScript = fc.getSelectedFile();
          if (!fwScript.getName().endsWith(".sh")) {
            fwScript = new File(fwScript + ".sh");
          }
          if (activePolicy.isModified()) {
            PolicyManager.getInstance().saveActivePolicy();
          }
          if (!activePolicy.isModified()) {
            IPTablesParser parser = new IPTablesParser();
            parser.generateScript(activePolicy,
activePolicy.getFirewalls()[0], fwScript);
            me.setStatus("Script saved to " + fwScript.getAbsolutePath());
          }
          else {
            logger.log(Level.WARNING,          "Policy:          "          +
activePolicy.getName() + " was not saved, so no script was created");
          }
        }
      }
    }
}
```

Policy Install

```
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.io.*;


public class PolicyInstall extends AbstractAction {
    private final JFrame frame;
    private Preference prefs =
PreferenceManagerInstance.getInstance().getPreference();
```

```
public PolicyInstall(JFrame frame) {
   super(PropertyManager.getString("menu.policy.install"));
   this.frame = frame;
}

public void actionPerformed(final ActionEvent e) {
   Policy activePolicy = PolicyManager.getInstance().getActivePolicy();
   ManagedFirewall[] firewalls = activePolicy.getFirewalls();
   Object[] list = new Object[firewalls.length];
   for (int i = 0; i < firewalls.length; i++) {
      list[i] = firewalls[i];
   }
        ObjectListDialog  old  =  new  ObjectListDialog(frame,  list,
PropertyManager.getString("gui.policy.choose"), true);
      Object[] selected = old.getSelectedObjects();

      TextViewer tv = new TextViewer(frame,
PropertyManager.getString("menu.policy.install"));

         for (int f = 0; f < selected.length; f++) {
       ManagedFirewall wall = (ManagedFirewall) selected[f];

         File fwScript = new File(prefs.getDirGeneratedScripts().toString() +
System.getProperty("file.separator") + wall.getName() + ".auto");
         File fwStopScript = new
File(prefs.getDirGeneratedScripts().toString() +
System.getProperty("file.separator") + wall.getName() + ".stop");
         File rcScript = new File(prefs.getDirGeneratedScripts().toString() +
System.getProperty("file.separator") + wall.getName() + ".rc");
         IPTablesParser parser = new IPTablesParser();
         parser.generateScript(activePolicy, wall, fwScript);
         parser.generateStopScript(activePolicy, wall, fwScript);
         parser.generateRCScript(wall, rcScript);
         Connection conn = wall.getConnection();
         tv.append("Authenticating...");
         conn.connect();
         tv.append("Transfering firewall script...");
         conn.put(fwScript.getAbsolutePath(), wall.getFWScriptPath(), 0700,
  wall.isUseSudo());
```

```
        tv.append("Creating rc script");
        conn.put(rcScript.getAbsolutePath(), wall.getRCScriptPath(), 0700,
wall.isUseSudo());
        tv.append("Creating symlinks...");
        InputStream in = conn.execute("/bin/ln -s " + wall.getRCScriptPath()
+ " /etc/rc3.d/S05jwall 2>&1\n", wall.isUseSudo());
        try {
          byte buffer[] = new byte[255];
          int read;
          while ((read = in.read(buffer)) > 0) {
            String out = new String(buffer, 0, read);
            tv.append(out);
          }
        }
        catch (IOException ioe) {
          System.out.println(ioe);
        }
        in  =  conn.execute("/bin/ln  -s  "  +  wall.getRCScriptPath()  +  "
/etc/rc5.d/S05jwall 2>&1\n", wall.isUseSudo());
        try {
          byte buffer[] = new byte[255];
          int read;
          while ((read = in.read(buffer)) > 0) {
            String out = new String(buffer, 0, read);
            tv.append(out);
          }
        }
        catch (IOException ioe) {
          System.out.println(ioe);
        }
        tv.append("Executing script...");
        in = conn.execute("/bin/sh " + wall.getFWScriptPath() +" 2>&1\n",
wall.isUseSudo());
        try {
          byte buffer[] = new byte[255];
          int read;
          while ((read = in.read(buffer)) > 0) {
            String out = new String(buffer, 0, read);
            tv.append(out);
          }
```

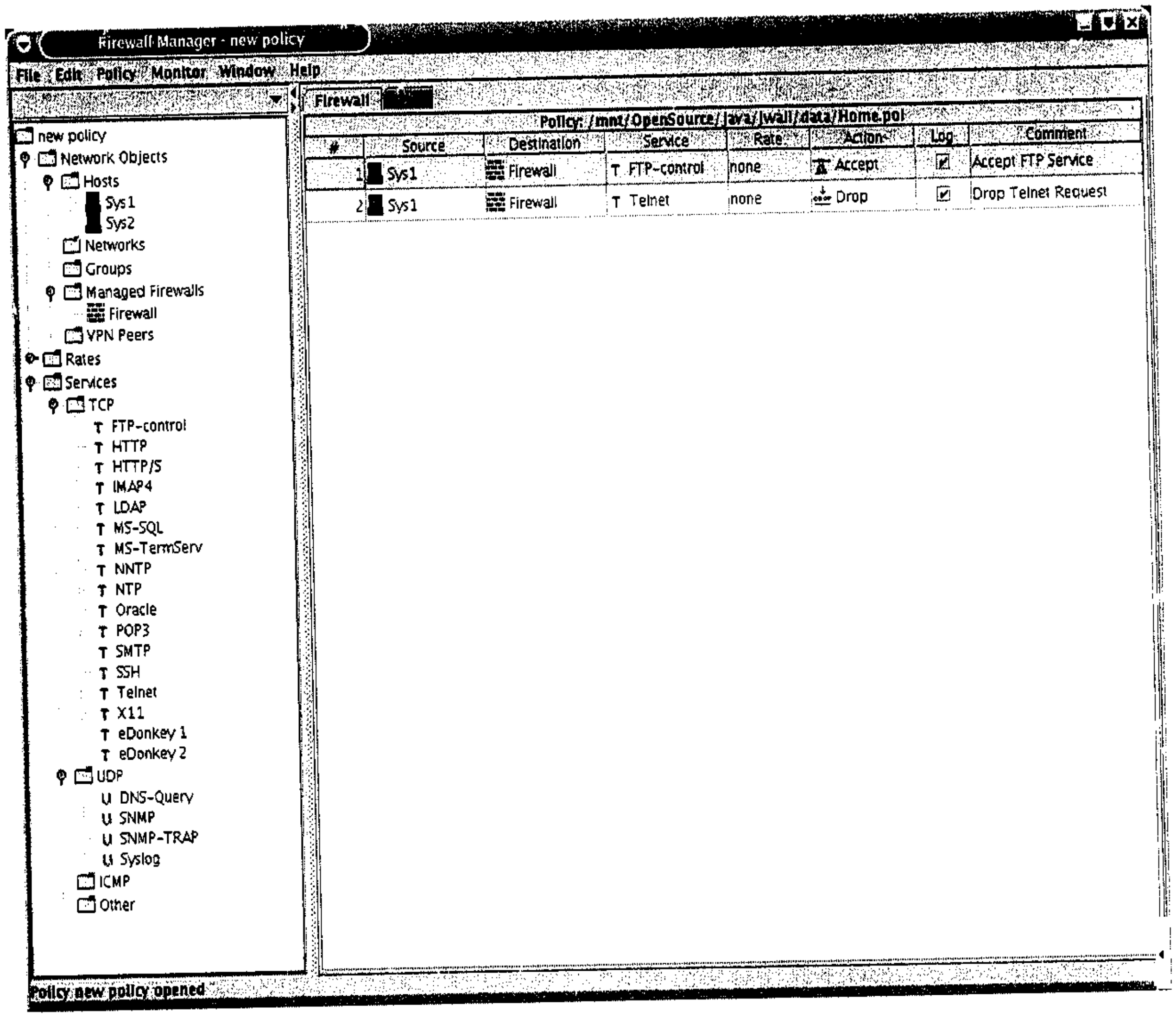
```
            }
            catch (IOException ioe) {
                System.out.println(ioe);
            }
            tv.append("Disconnecting...");
            conn.disconnect();
            tv.append("Complete.");
        }
    }
}
```

Policy Uninstall

```
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.io.IOException;
import java.io.InputStream;


public class PolicyUninstall extends AbstractAction {
    private final JFrame frame;


    public PolicyUninstall(JFrame frame) {
        super(PropertyManager.getString("menu.policy.uninstall"));
        this.frame = frame;
    }

    public void actionPerformed(final ActionEvent e) {
        Policy activePolicy = PolicyManager.getInstance().getActivePolicy();
        ManagedFirewall[] firewalls = activePolicy.getFirewalls();
        Object[] list = new Object[firewalls.length];
        for (int i = 0; i < firewalls.length; i++) {
            list[i] = firewalls[i];
        }
        ObjectListDialog    old    =    new    ObjectListDialog(frame,    list,
PropertyManager.getString("gui.policy.choose"), true);
        Object[] selected = old.getSelectedObjects();
    TextViewer tv = new TextViewer(frame,
PropertyManager.getString("menu.policy.uninstall"));


    for (int f = 0; f < selected.length; f++) {
      ManagedFirewall wall = (ManagedFirewall) selected[f];

      Connection conn = wall.getConnection();
      tv.append("Authenticating...");
      conn.connect();
      tv.append("Stopping firewall...");
      InputStream in = conn.execute(wall.getRCScriptPath() + " stop
2>&1\n", wall.isUseSudo());
      try {
        byte buffer[] = new byte[255];
        int read;
        while ((read = in.read(buffer)) > 0) {
          String out = new String(buffer, 0, read);
          tv.append(out);
        }
      }
      catch (IOException ioe) {
        System.out.println(ioe);
      }
    }
  }
}
```

APPENDIX 2

Adding New Rules


Firewall Manager - new policy
File Edit Policy Monitor Window Help
Firewall

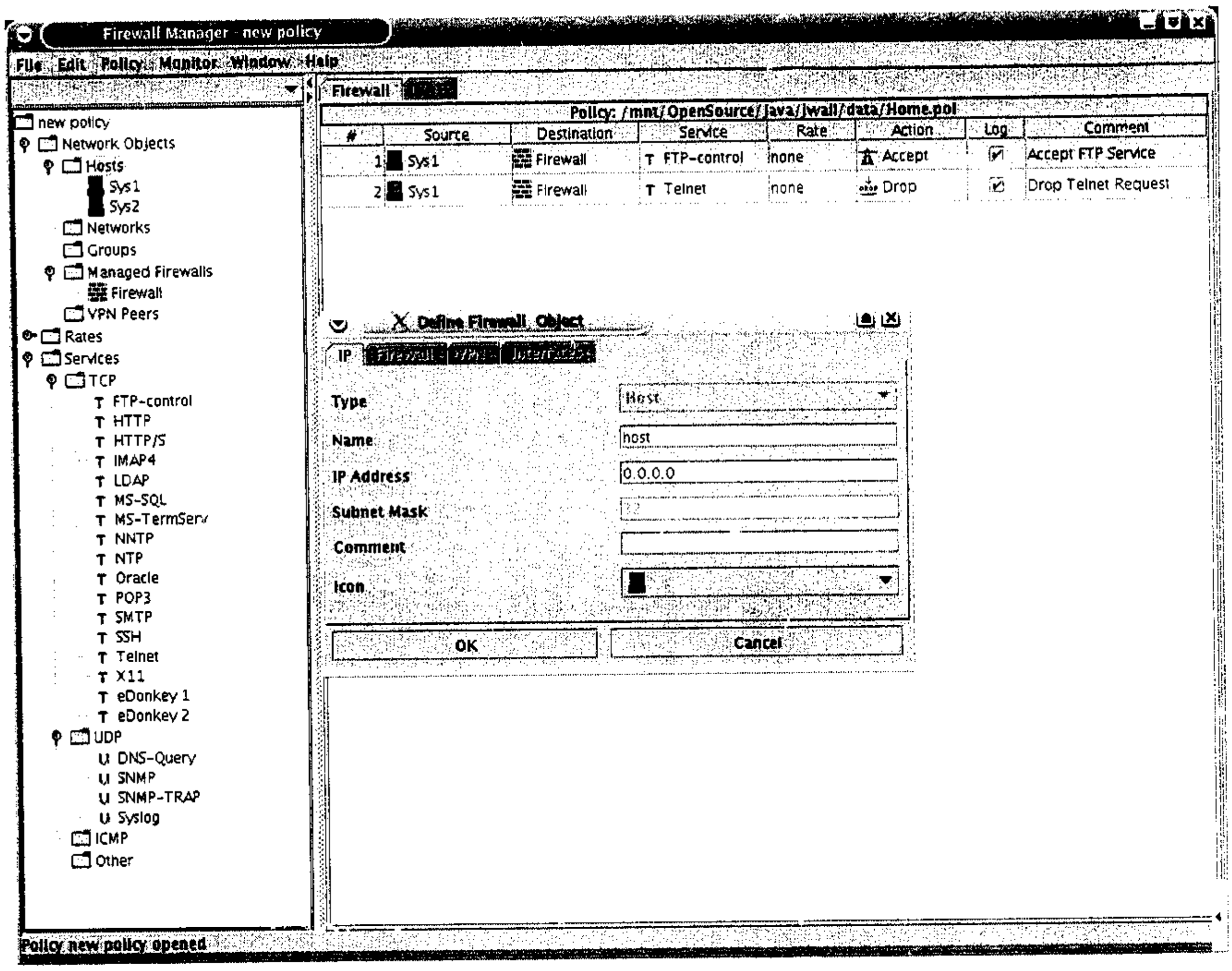
Policy: /mnt/OpenSource/java/jwall/data/Home.pol

#	Source	Destination	Service	Rate	Action	Log	Comment
1	Sys1	Firewall	FTP-control	none	Accept	✓	Accept FTP Service
2	Sys1	Firewall	Telnet	none	Drop	✓	Drop Telnet Request

```
new policy
  Network Objects
    Hosts
      Sys1
      Sys2
    Networks
    Groups
    Managed Firewalls
      Firewall
    VPN Peers
  Rates
  Services
    TCP
      T FTP-control
      T HTTP
      T HTTP/S
      T IMAP4
      T LDAP
      T MS-SQL
      T MS-TermServ
      T NNTP
      T NTP
      T Oracle
      T POP3
      T SMTP
      T SSH
      T Telnet
      T X11
      T eDonkey 1
      T eDonkey 2
    UDP
      U DNS-Query
      U SNMP
      U SNMP-TRAP
      U Syslog
    ICMP
    Other
```

Policy new policy opened

Define Host Object



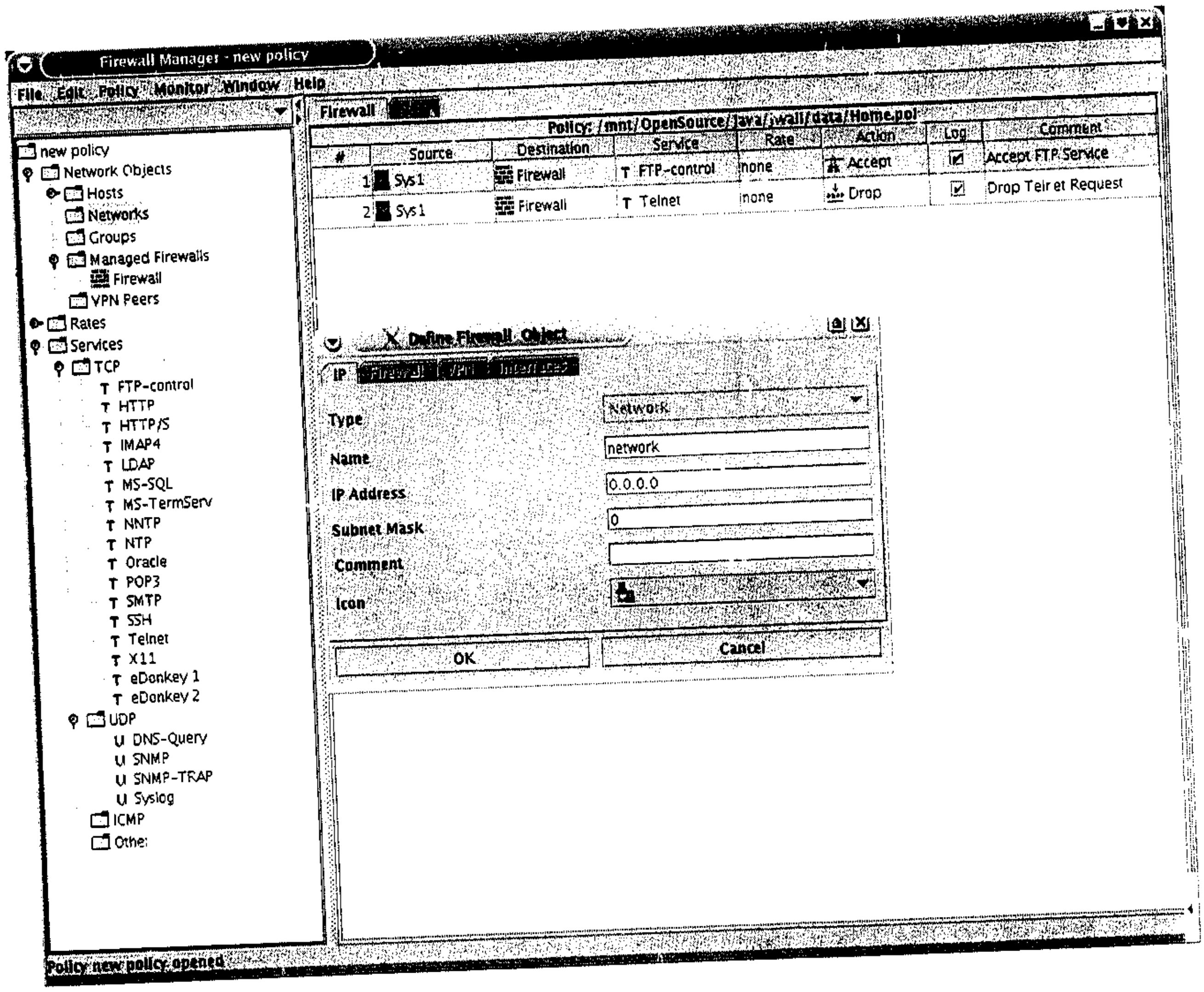
Policy: /mnt/OpenSource/Java/Jwall/data/Home.pol

#	Source	Destination	Service	Rate	Action	Log	Comment
1	Sys1	Firewall	FTP-control	none	Accept	✓	Accept FTP Service
2	Sys1	Firewall	Telnet	none	Drop	✓	Drop Telnet Request

Type	Host
Name	host
IP Address	0.0.0.0
Subnet Mask	32
Comment	
Icon	■

Policy new policy opened

Define Network Object



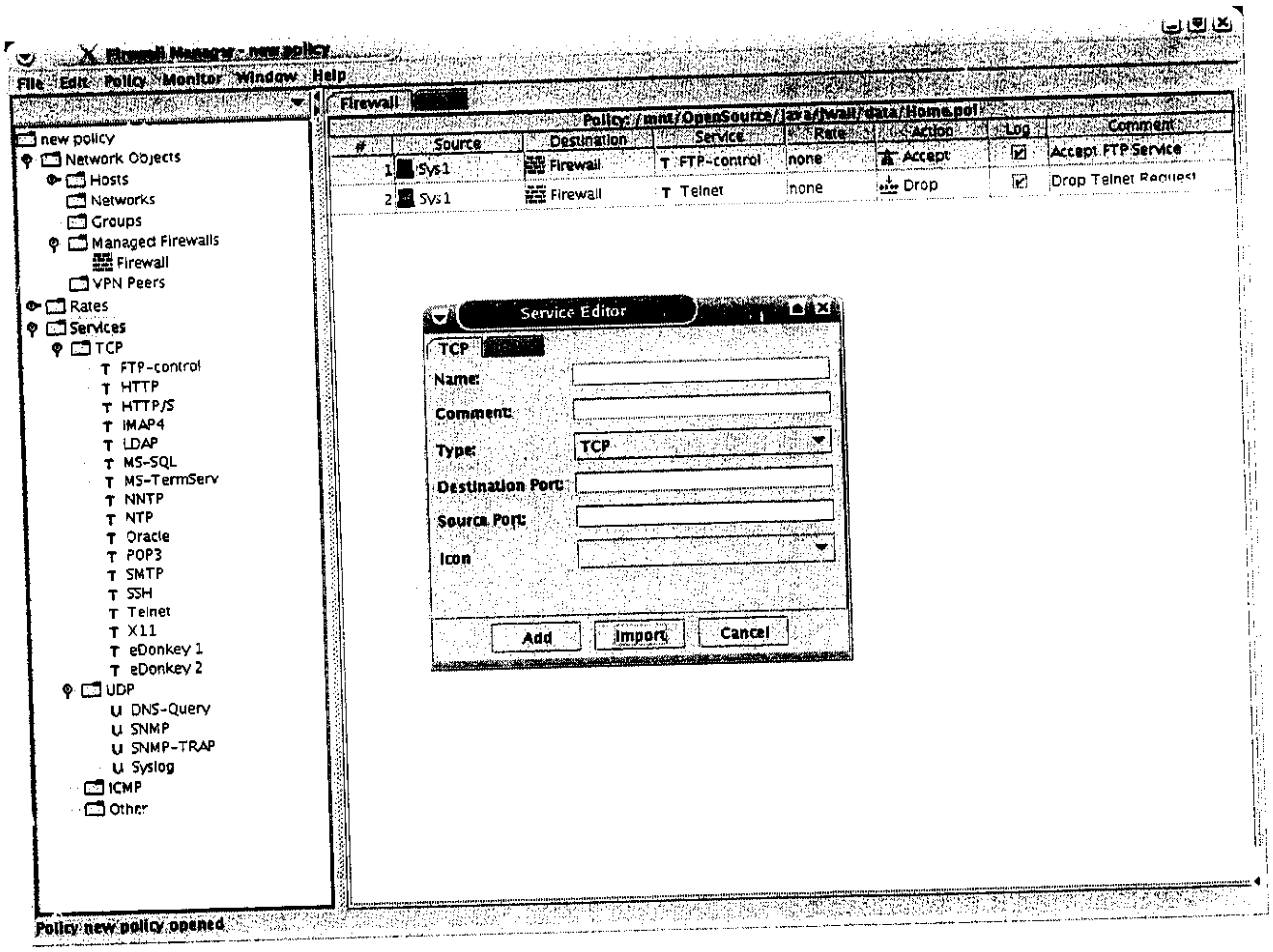
Policy: /mnt/OpenSource/java/jwall/data/Home.pol

#	Source	Destination	Service	Rate	Action	Log	Comment
1	Sys1	Firewall	FTP-control	none	Accept	✓	Accept FTP Service
2	Sys1	Firewall	Telnet	none	Drop	✓	Drop Telnet Request

Type	Network
Name	network
IP Address	0.0.0.0
Subnet Mask	0
Comment	
Icon	

Policy new policy opened

Defining services


Firewall Manager - new policy

File Edit Policy Monitor Window Help

- new policy
  - Network Objects
    - Hosts
    - Networks
    - Groups
    - Managed Firewalls
      - Firewall
    - VPN Peers
  - Rates
  - Services
    - TCP
      - T FTP-control
      - T HTTP
      - T HTTP/S
      - T IMAP4
      - T LDAP
      - T MS-SQL
      - T MS-TermServ
      - T NNTP
      - T NTP
      - T Oracle
      - T POP3
      - T SMTP
      - T SSH
      - T Telnet
      - T X11
      - T eDonkey 1
      - T eDonkey 2
    - UDP
      - U DNS-Query
      - U SNMP
      - U SNMP-TRAP
      - U Syslog
    - ICMP
    - Other

Firewall

Policy: /mnt/OpenSource/java/jwall/data/Home.pol

#	Source	Destination	Service	Rate	Action	Log	Comment
1	Sys1	Firewall	T FTP-control	none	Accept	✓	Accept FTP Service
2	Sys1	Firewall	T Telnet	none	Drop	✓	Drop Telnet Request

Service Editor

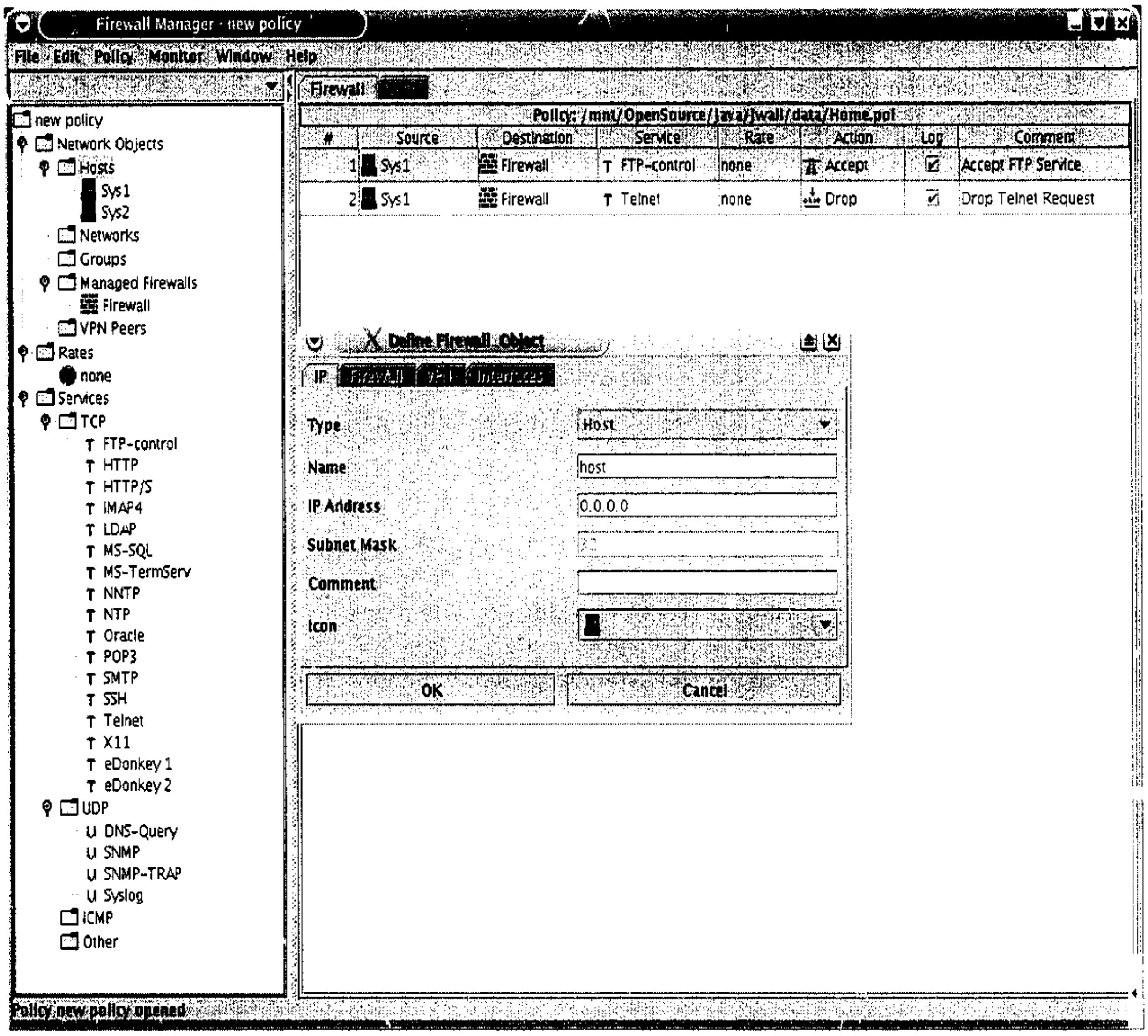
TCP

Name:	
Comment:	
Type:	TCP
Destination Port:	
Source Port:	
Icon	

Add Import Cancel

Policy new policy opened

Defining Firewall Object



Policy: /mnt/OpenSource/java/jwall/data/Home.pol

#	Source	Destination	Service	Rate	Action	Log	Comment
1	Sys1	Firewall	FTP-control	none	Accept	✓	Accept FTP Service
2	Sys1	Firewall	Telnet	none	Drop	✓	Drop Telnet Request

Type	Host
Name	host
IP Address	0.0.0.0
Subnet Mask	32
Comment	
Icon	

Policy new policy opened

REFERENCES

1. Cay S. Horstmann, Gary Cornell 'CORE JAVA 2 Vol-I fundamentals', Sun Microsystem Press, 2003 .

2. Cay S. Horstmann, Gary Cornell (2001) 'CORE JAVA 2 Vol-II fundamentals', Sun Microsystem Press, 2003 .

3. H.M. Deitel & V.M. Deitel (2002) 'JAVA-How to Program', Tata McGraw Hill Publications, 2002.

4. www.linux.org

5. www.linuxdoc.org

6. www.netfilter.org

7. www.sun.com