INTEGRATION OF NETWORK SERVICES - PFD

PROJECT REPORT P- 1212

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT
FOR THE AWARD OF THE DEGREE

BACHELOR OF COMPUTER SCIENCE AND ENGINEERING
OF
BHARATHIAR UNIVERSITY, COIMBATORE.

Submitted by,

P. KALEESWARAN (0027KO174)

G.KISHORE PRASADH (0027K1116)

Guided by,

Ms. S. RAJINI B.E.,

Senior Lecturer, Department of Computer Science and Engineering.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KUMARAGURU COLLEGE OF TECHNOLOGY
(Affiliated to Bharathiar University, Coimbatore)
COIMBATORE – 641006.

APRIL 2004.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY
(Affiliated to Bharathiar University, Coimbatore)

CERTIFICATE

This is to certify that the project entitled

INTEGRATION OF NETWORK SERVICES- PFD
is done by

P.Kaleeswaran
0027KO174

G.Kishore Prasadh
0027K1116

and submitted in partial fulfillment of the requirement
for the award of the degree of the

Bachelor of Computer Science and Engineering
Of
Bharathiar University, Coimbatore.

[illegible]
Professor & Head of the department
(Dr. S. THANGASAMY)

[illegible]
Guide
(Ms. S. RAJINI)

Certified that the candidate was examined by us in the Project work And Viva - voce examination held on 23-03-04

[illegible] 24/3/'04
Internal Examiner

External Examiner

DECLARATION

We,

P.KALEESWARAN (0027KO174)

G.KISHORE PRASADH (0027K1116)

declare that the project entitled "INTEGRATION OF NETWORK SERVICES - PFD" is done by us and to the best of our knowledge, a similar work has not been submitted to the Bharathiar University or any other institution, for fulfillment of the requirement of the course study.

The report is submitted as partial fulfillment of the requirement for the award of the degree of Bachelor of Computer Science and Engineering of Bharathiar University.

[illegible]
P.KALEESWARAN

[illegible]
G.KISHORE PRASADH

Countersigned:

[illegible]

GUIDE: Ms. S. RAJINI B.E.,
Senior Lecturer, Department of Computer Science and Engineering,
Kumaraguru College of Technology,
Coimbatore – 641006.

Place : KCT, COIMBATORE

Date : 23-03-04.

ACKNOWLEDGEMENT

We would like to begin with a special note of gratitude to our Principal **Dr.K.K.Padmanaban, B.Sc. (Engg), M.Tech, Ph.D.**, and our Head of Department **Prof.Dr.S.Thangaswamy, Ph.D.** for their encouragement and making available to us the much needed facilities for successfully completing this project.

We also take immense pleasure in thanking our project coordinator **Mrs.D.Chandrakala, M.E, Senior Lecturer,** without whose encouragement our endeavor would not have been successful.

We are eternally indebted to our project guide **Ms.S.Rajini B.E, Senior Lecturer**, who has been a constant source of inspiration and guidance. Her gentle yet firm goading helped us finish our project on time

Our sincere thanks to our parents, all the staff members and our dear friends who all played small but important parts in helping us complete this project.

Above all, we owe our gratitude to the Almighty, for showering abundant blessings on us.

SYNOPSIS

This project deals with integrating the configuration of available network services like Proxy server, Firewall and DNS (Domain Name System) and thus the name of our project **INTEGRATION OF NETWORK SERVICES - PFD**

Proxy server sits between the client program (web browser) and external server (another server on the web) to filter requests, improve performance and share connections. The configuration of proxy involves making changes in the squid.conf file. The particulars that are required for the configuration of proxy is given via the interface are written and the respective changes are made in the squid.conf file. The configuration of proxy here mainly involves restricting the sites that are to be surfed, restricting certain search words, IP restriction and controlling the number of sessions.

Firewall is used to secure the Internet transactions. A firewall prevents any direct unauthorized attempt at access. The functions that are intended to be provided are enhanced packet filtering, proxy services etc. Similar to the proxy interface, the firewall interface also gets the required details and the respective rules are written in script file using the '"ipchain".

The Domain Name System is a distributed Internet directory service. DNS is used mostly to translate between domain name and IP address. In this interface facilities are provided for adding a zone, viewing the existing zones and adding specific records to an already existing zone and removing a zone. The interface gets all the information regarding the configuration and the necessary changes are made in the corresponding files.

CONTENTS

1. INTRODUCTION

1.1 Existing system and its limitations

The Proxy, Firewall and the DNS in the existing Linux operating system can be configured only if we have a detailed knowledge about the configuration files that are dealt with it. To configure proxy, access control lists are to be written in the Squid.conf file. Writing the ACLs in the squid.conf file needs knowledge about the various methodologies of writing ACLs and the exact locations of writing those.

When the DNS is taken into consideration it contains many record files such as the A record, SOA record, MX record etc. When a DNS is to be configured information are needed to be entered into these records, which again needs a clear knowledge. Also details of writing the code for including the domain name in the named.conf file are also required.

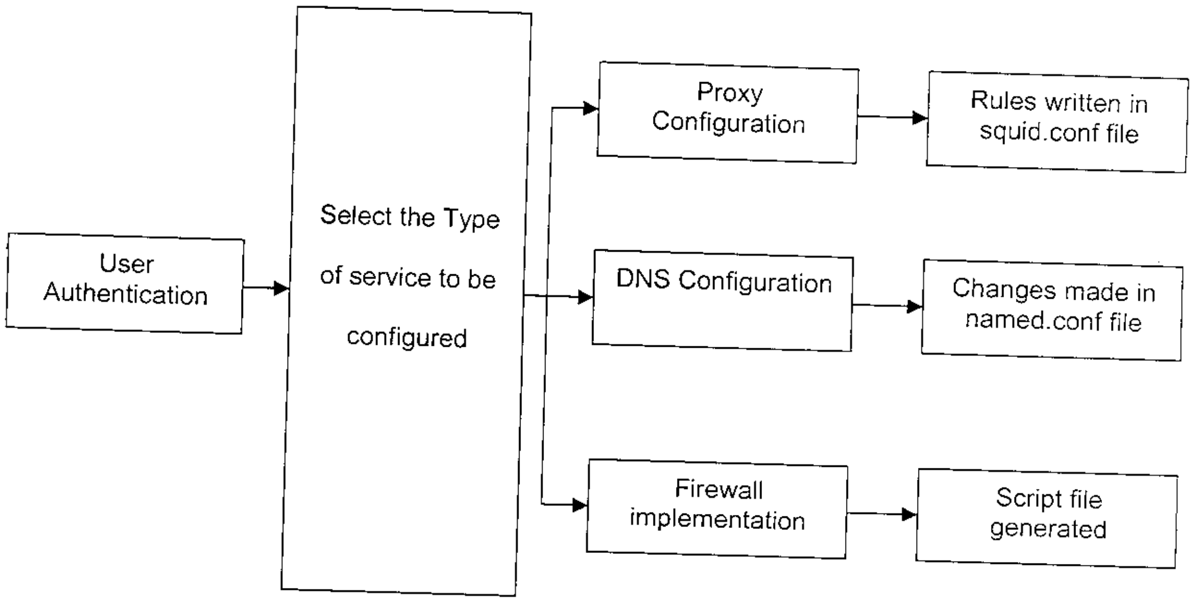
To implement a Firewall a series of rules have to be written, to govern what kind of access we want to allow on a system. Here again the awareness for writing rules is necessary.

1.2 Proposed system overview and its advantages

In the existing system we configure Proxy, Firewall and DNS separately. In such case the administrators need to have complete knowledge about each and every configuration file. For example for configuring the proxy the administrator need to know the several options available in the squid.conf file. Also the manual configuration takes so much of time. Our interface provides a way to configure such configuration files automatically. The configuration includes denying the various timely unwanted options.

The interface provides automatic configuration of the various configuration files. That is, it is enough if the administrator enters the details regarding the configuration via the interface. The changes in the corresponding file are made automatically. In the case of proxy, for URL restriction it is enough for the administrator to specify the name of the URL alone in the interface, which will write the corresponding access control list in the squid.conf file. Similarly the other details for the configuration of the proxy is got via the interface and the necessary changes are made in the corresponding files.

In the case of configuring DNS the detail such as the name of the zone that is to be added, the changes that have to be made in the existing zone etc. are entered in the interface, which makes the corresponding changes in the respective file. Similarly getting the necessary details via the interface does the configuration of firewall and the corresponding script is generated. The proposed system is found to have three main limitations such as platform dependence.


User Authentication
Select the Type of service to be configured
Proxy Configuration
Rules written in squid.conf file
DNS Configuration
Changes made in named.conf file
Firewall implementation
Script file generated

BLOCK DIAGRAM OF THE PROJECT

2. SOFTWARE REQUIREMENTS ANALYSIS

2.1 Project Definition.

The PFD Joint Effort service includes an interface via which it is possible to configure Proxy, Firewall and DNS. The core objective of our project is to reduce the work of an administrator in configuring the Internet services like Proxy, Firewall and DNS.

It is a very intricate work for an administrator to configure this security service without in-depth knowledge about the files that are to be configured. Each and every security service has certain files to be configured. Detail studies of these files are necessary before we configure them. Our project reduces all these work of an administrator and provides him a friendly interface to implement the security service

2.2 Project plan

In the analysis phase, requirements for configuring the network services Proxy, Firewall, and DNS was discussed and concepts of Linux were studied.

Next the system design was established where the complete system was depicted in the form of event flow diagram.

In the Implementation phase, all the necessary coding were written to split the overloaded application, process each file and merge the processed contents.

In the testing phase, all the necessary tests were performed to test the validity of the code developed.

3. SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Purpose

The PFD Joint Effort service includes an interface via which it is possible to configure Proxy, Firewall and DNS. The interface reduces the effort of an administrator in configuring the above security services, by relaxing his in-depth knowledge regarding the configuration

3.2 Scope

We create an interface through which an administrator can configure the three Internet services mentioned above. Here the administrator does not require the knowledge about the changes that have to be made in the corresponding configuration files. The required input is got via the interface and the necessary changes are made automatically in the needed configuration files. The interface is designed in such a way that it is very easy for the administrator to configure the Internet service.

3.3 Product Overview and Summery

Our project takes in the essential details for the configuration of the Proxy, Firewall and DNS and makes the necessary actions that are required for the configuration. Initially we authenticate the administrator by means of password checking. If he is a valid user then he is asked to select the service he wants to configure. If his option is to configure Proxy then we enable him to perform four activities such as URL restriction, search word restriction, IP filtering and controlling number of sessions.

After getting the required information the configuration is done in the Squid file. For the configuration of DNS we ask for the following options such as whether to add a zone, to modify an existing zone or to remove a zone. After taking the necessary information corresponding changes are made in the respective files. For the configuration of Firewall the details such as the source, destination, protocol, port, log and the action to be performed are extracted from the administrator via the interface and the corresponding script file is generated.

3.3.1 Introduction to PHP

All the interfacing is done using the server side scripting language PHP (Hypertext Pre Processor). PHP is one of the scripting languages that have emerged as a world leader. PHP is a server-side scripting language that is platform independent. It is used to create dynamically generated Web pages quickly. It can also be used to create database-driven Web sites for e-commerce, community portals, and other Web-based applications. Today, PHP is probably the most widely used non-Microsoft scripting language. PHP can be deployed over windows platforms running IIS and SQL and over Linux platforms running Apache Web server and MySQL

Features of PHP

- PHP is a server-side scripting language.
- Capable of handling database-driven e-commerce Web sites.
- Easer to write than any other scripting language.
- Improved with every release.
- Embedded within HTML to create dynamic and interactive Web pages.
- Used to create images.
- Is used to read and write files.
- Is used for sending emails.

3.3.2 Introduction to Linux

Linux is a modern operating system. This means that it runs on 32 bit architecture, uses preemptive multitasking, protected memory, supports multiple users, and has a rich support for networking, including TCP/IP networking. Linux was originally written for Intel's 386 architecture, but now runs on a wide variety of hardware, including the full x86 family of processor, as well as Alpha, SPARC and Power PC chips.

Linux runs on all the applications a UNIX server system should run, including web servers like APACHE, mail serving software like send mail, and database servers like ORACLE, Informix, or more open applications like my SQL and postgres. Linux supports a wide range of file system types, and through programs like SAMBA can even seamlessly replace Windows NT as a windows fileserver.

Technically speaking, Linux is a Kernel, the core part of an operating system that handles networking, hardware management, and basically makes the whole things run. Red hat Linux is currently the most popular Linux distribution. Red hat maintains an extensive library of Linux documentation

Linux Features

- Multi-user Operating system
- Linux is a free GPL Licensed
- The freedom to make improvements to he program, and to make those improvements public
- The freedom to examine and study the source code of the program and to change it to suite our needs

3.3.3 Introduction to Apache Server

The Apache Web server is a full-featured free HTTP Web server. The server software includes server daemon, configuration files, management tool, and documentation. Most distributions, such as Red Hat, Caldera, and SUSE, are provided with the option of installing the Apache Web server during the installation of Linux system.

All the web pages are placed in */var/www/html* directory or in any sub directory. The system is already configured to operate as a Web server. What we have to perform is to do any needed network server configurations, and then designate the files and directories open to remote users. Once the Web site is connected to a network, remote users can access it.

3.4 Development And Operating Environment

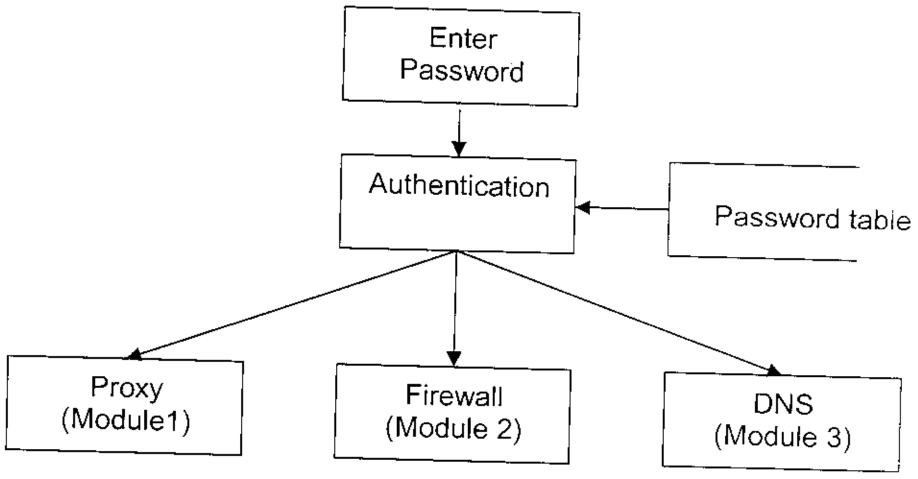
Hardware Description

RAM	:	64MB SDRAM
Cache	:	512 KB cache
Floppy Drive	:	1.44 MB FDD
Hard Drive	:	10.2 GB HDD
Monitor	:	15" Digital Color Monitor
Keyboard	:	107 Keys Mechanical keyboard
Mouse	:	PS2 Mouse

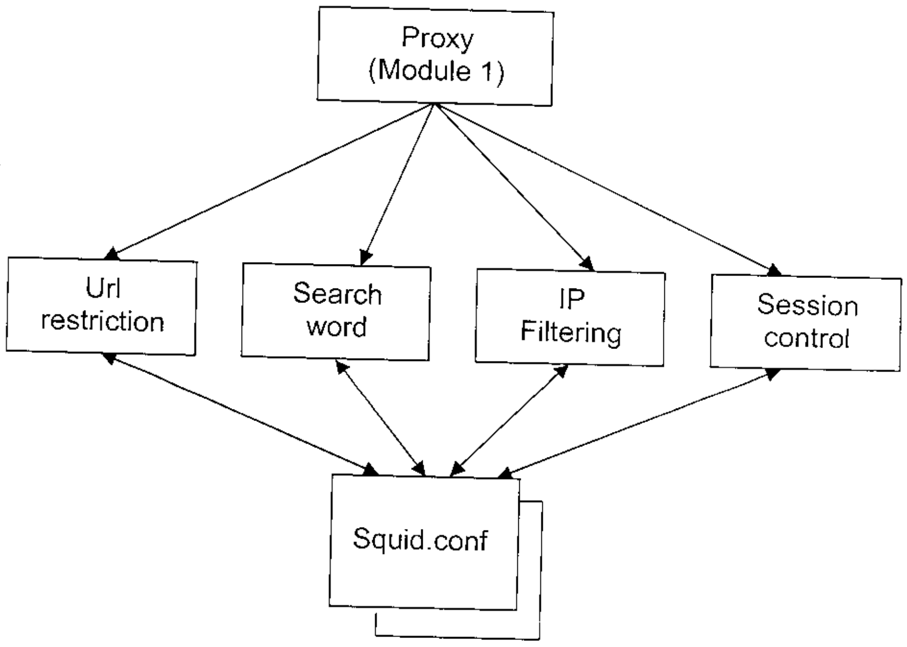
Software Description

Operating System	:	Linux Red Hat 7.2
Software	:	PHP 4.1
Browser	:	Netscape/ Mozilla
Web Server	:	Apache Server

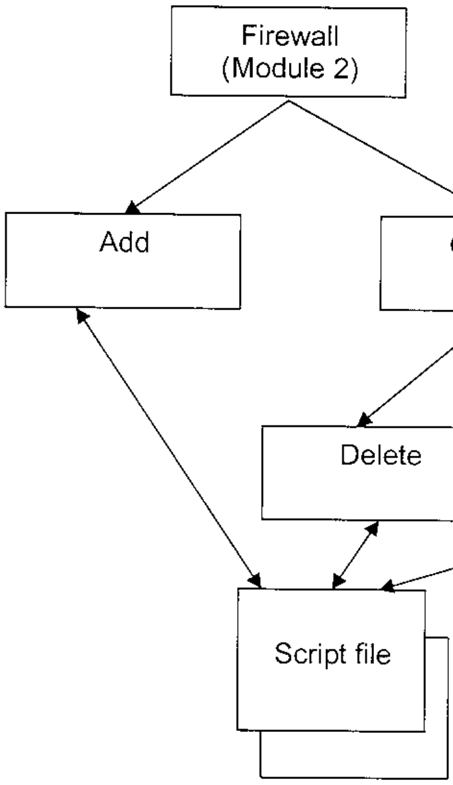
3.5 Data Flow Diagrams



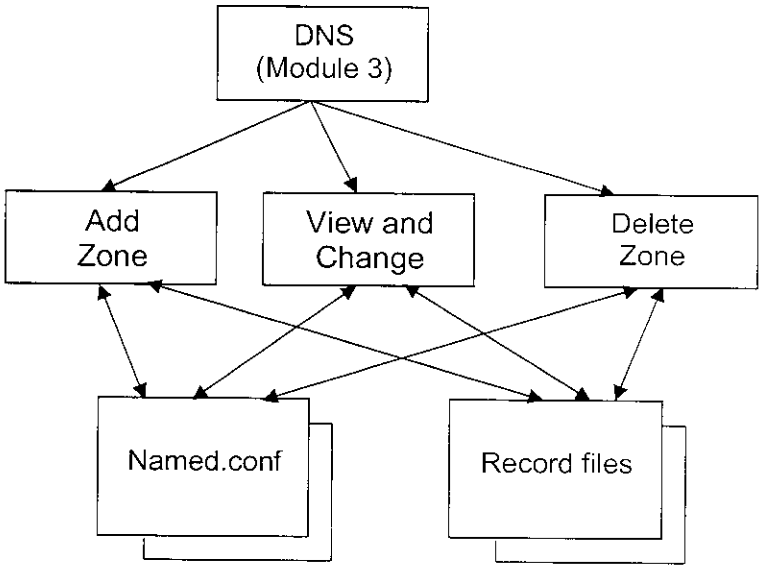
DFD FOR PROXY



DFD FOR FIREWALL



DFD FOR DNS

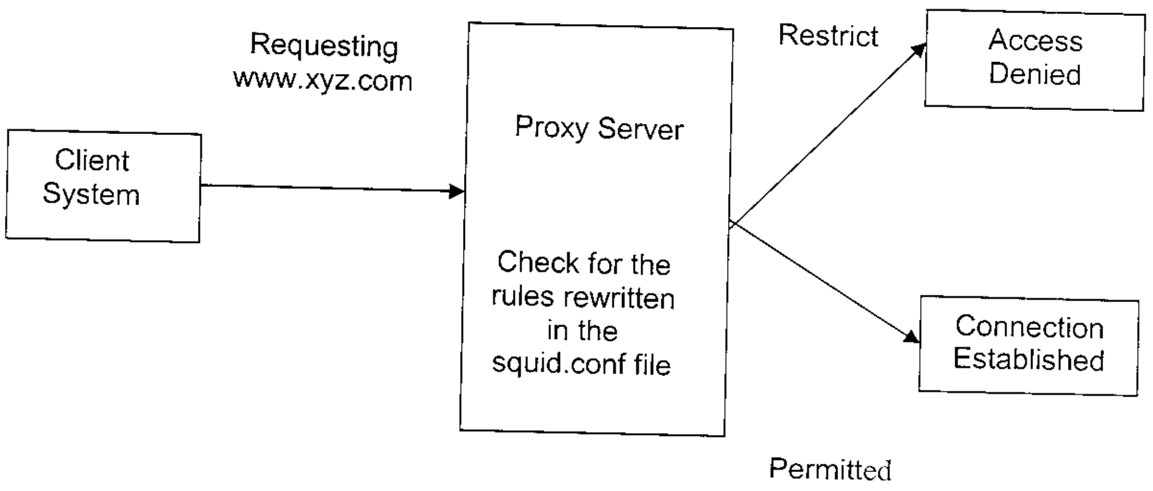


Introduction

The three modules involved in our project are creating interface for the configuration of proxy, firewall and DNS, which this section describes in detail.

Proxy Server - Squid

Proxy server is defined as a server that provides Internet access to different users at the same time i.e., by sharing a single Internet connection. A good proxy server also provides for caching of the requests, which helps to access data from local resources rather fetching the data from web thus reducing access time and bandwidth. Squid is one such software, which supports proxy, caching of HTTP, ftp, gopher, etc. It also supports SSL, access controls, caching of DNS and maintains a full log of the entire request.



CONNECTION MADE TO A PROXY SERVER

Configuring the proxy involves configuring the Squid. Squid is a proxy-caching server for web clients, designed to speed Internet access and provide security controls for web servers. It implements a proxy-caching service for web clients that caches web pages as users make requests. Copies of web pages accessed by users are kept in the squid cache and as request are made, Squid checks to see if it has a current copy. If squid have a current copy, it returns the copy from its cache instead of querying the original site.

Starting Squid

By default, squid will not give any access to clients and access controls have to be modified for this purpose. After listing out the rules squid has to be started using the following command

```
service squid start
```

Squid can be started automatically at boot time by enabling it in *ntsysv* or setup (System Service Menu). After each and every change in the configuration file, the present squid process has to be stopped and for the new configuration changes to take effect squid have to be started once again. These two steps can be achieved by the following command

```
service squid restart or
/etc/rc.d/init.d/squid restart
```

Squid Configuration

The squid configuration file is squid.conf, located in the /etc/squid directory. The first step in configuring Squid security is to create ACLs. These are lists of hosts and domains for which we want to set up control. We define ACL's using the acl command, with which we create labels for the systems on which we are setting controls. We then use commands, such as http_access, to define these controls. We can define a system, or a group systems, based on several acl options, such as the source IP address, the domain name, or even the time and date.

Example

For example, the src option is used to define a system or a group of systems with a certain source address. To define a mylan acl entry for the systems in local network with the addresses 192.169.1.0 through to 192.168.1.255,use the following ACL definition:

Acl mylan src 192.168.1.0/255.255.255.0

Once defined, we can use an ACL definition in Squid option to specify a control we want to place on those systems. For example, to allow access by the mylan group of local systems to the web through the proxy, use a http_access option with the allow action specifying mylan as the acl definition to use, as shown here.

http_access allow mylan

By defining ACLs and using them in Squid options, we can tailor our website with the kind of security we want. The following example allows access to the web through the proxy by only the mylan group of local systems, denying access to all others. Two acl entries are set up, one for the local system and one for all others. http_access options first allow access to the local system, and then deny access to all others.

```
Acl mylan src 192.168.1.0/255.255.255.0
Acl allsystems src 0.0.0.0/0.0.0.0
http_access allow mylan
http_access deny all systems
```

The order of the http_access options is important. Squid starts from the first and works down, stopping at the first http_access option with an ACL entry that matches.

The some of the options available in squid are:

Squid ACL options

S.No	Option	Description
1.	**Src** ip-address/netmask	Client's IP address
2.	**Src** addr1-addr2/netmask	Range of IP address
3.	**Dst** ip-address/netmask	Destination IP address
4.	**Myip** ip-address/netmask	Local socket IP address
5.	**url-regex [-i]** expression	Regular expression matching on whole URL
6.	**Urlpath_regex [-i]** expression	Regular expression matching on URL path
7.	**Maxconn** number	Number of sessions to be restricted

After making changes to the squid.conf file we have to restart the service using the following command.

```
Service squid restart
```

Some of the squid options that we have taken into considerations are as follows:

1.URL restriction

URL restriction deals with restricting particular sites from being accessed by the client. For example sites such as movies.com can be restricted.

2.Search word restriction

Search word restriction deals with restricting connections whose URLs contain some specific words. For example words such as chat.

3.IP restriction

IP restriction is restricting clients with a specific IP address. The different forms of specifying the IP address are

- Clients IP address / subnet mask
- Range of addresses / subnet mask
- Host's IP address

4.Session restriction

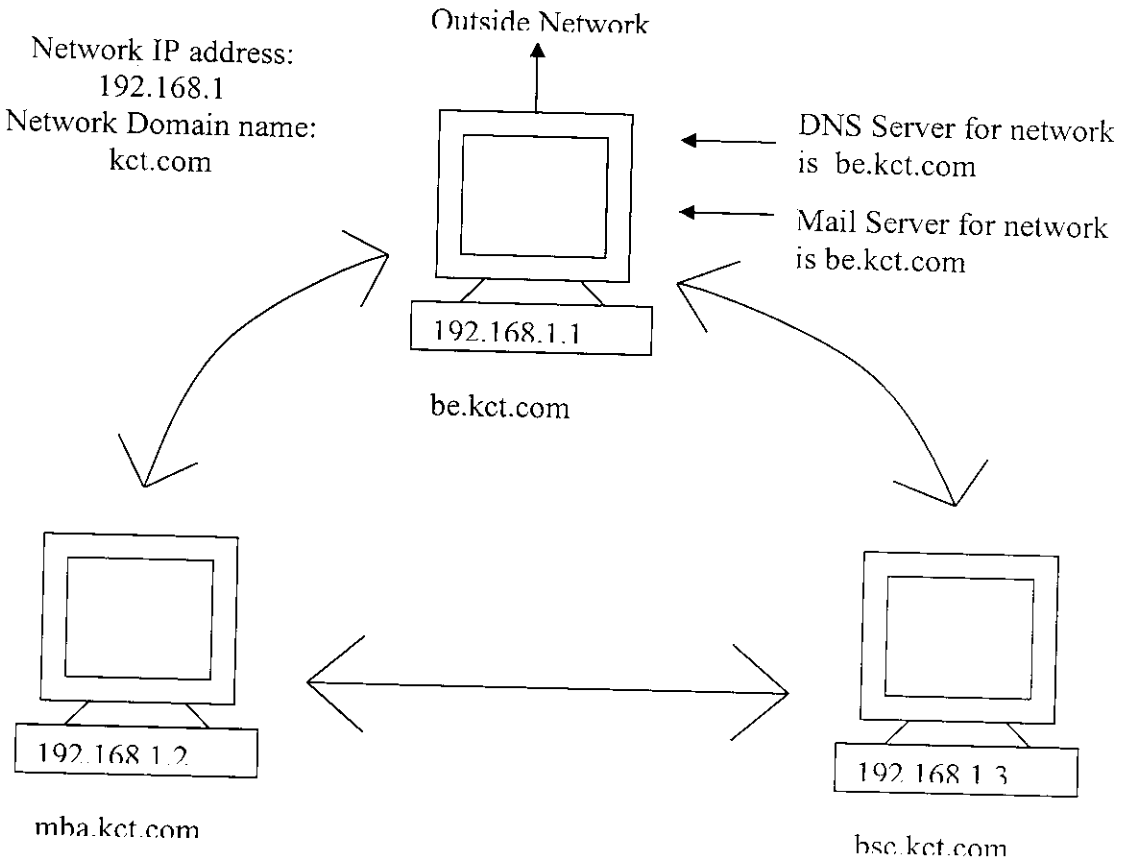
Session restriction deals with limiting the number of HTTP connections established at a time. For example number of sessions can be limited to 5.

Domain Name System

The Domain Name System (DNS) is an Internet service that converts domain names into their corresponding IP address. All computers connected to the Internet are addressed using an Internet Protocol (IP) address. This consists of a number composed of four segments separated by periods. Depending on the type of network, several of the first segments are used for the network address and several of the last segments are used for the host address.

If we have three machines on our network, we can give then the address 192.168.1.1,192.168.1.2,192.168.1.3.We can then set up domain name services for our network by running a DNS server. We can then give our machine fully qualified domain names and configure our DNS servers to translate the names to their corresponding IP address. As shown in the figure below we could give the machine 192.168.1.1 the name be.sona.com and the machine 192.168.1.2 the name mba.kct.com.We can also implement Internet services on our network such as FTP, Web and mail services by setting up services for that on our network.

We can then configure our DNS server to let users access those services using fully qualified domain names. For example sona.com network the web server would be accessed using the name www.kct.com. Instead of domain name services we could have the /etc/hosts files in each machine contain the entire list of IP addresses and domain names for all the machines in our network. But, for any changes, we have to update each machine's /etc/hosts file.


Outside Network
Network IP address: 192.168.1
Network Domain name: kct.com
DNS Server for network is be.kct.com
Mail Server for network is be.kct.com
192.168.1.1
be.kct.com
192.168.1.2
mba.kct.com
192.168.1.3
bsc.kct.com

DNS NAME SERVERS

Zone Files

We can configure DNS server using a configuration file, several zone files, and a cache file. The part of a network for which the name server is responsible is called a zone. A zone is not the same as domain, because in a large domain, we could have several zones, each with its own server. We could also have several zones, each with its own name server. We could also have one name server service several zones. In this case, each zone has its own zone file. The zone files hold resource records that provide host name and IP address associations for computers on the network for which the DNS server is responsible. Zone files exist for the server's network and the local machine.

The most commonly used zone types are described here.

- Master zone

  This is the primary zone file for a network. It holds the mappings from domain names to IP address for all the hosts on the network.

- Slave zone

  These are references to other DNS servers for our network. Our network can have a master DNS server and several DNS servers to help carry the workload. A slave DNS server automatically copies its configuration files, including all zone files, from the master DNS server. Any changes to the master configuration files trigger an automatic download of these files to the slave servers. In effect, we have to manage the configuration files for the master DNS server, as they are automatically copied to the salve servers.

- . Zone

  This zone entry defines a hint zone specifying the root name servers.

DNS Servers

There are several kinds of DNS server, each designed to perform a different type of task under the domain name system. The basic kind of DNS server is the master server. Each network must have at least one master server that is responsible for resolving names on the network. Large networks may need several DNS servers. Some of these can be slave servers that can be updated directly from a master server. Others may be alternative master servers that hosts in a network can use. Both are commonly referred to as secondary servers.

For DNS request a DNS server cannot resolve, the requested can be forwarded to specific DNS server, outside the network, say on the Internet. DNS server in a network can set up to perform this task and are referred to as Forwarded servers. To help bear the workload, local DNS server can be setup within a network that operate as caching servers. Such a server merely collects DNS lookups from previous requests it sent to the main DNS server. The caching server can then answer any repeated requests.

A server that can answer DNS queries for a given zone with authority is known as an authoritative server, an authoritative server hold the DNS configuration records for hosts in a zone that will associate each host's DNS name with an IP address. For example a master server is an authoritative server. So are slave and stealth servers. A caching server is not authoritative server. It holds only whatever associations it picked up from other servers and cannot guarantee that the associations are valid.

DNS Configuration

As an alternative to making entries in the configuration files manually, we can use the Linuxconf DNS server configuration panels. We can use the dnsconf command to start up Linuxconf to perform only the domain name configuration tasks. This command displays a window with buttons for the different DNS server configuration panels. If we use the main Linuxconf window, select the Domain name server entry in the server's tasks list located under networking. Linuxconf provides panels for primary (master), secondary (slave), reverse mapping, and forwarder zone entries. We can also configure features such as access control and logging. When we finish configuration, Linuxconf generates new named.conf and zone files, use the domain panel to create zone files, and the reverse mapping panel to create corresponding reverse mapping zone files. In this panel, be sure to use only the network part of the IP address for the network number.

Configuration file - Named.conf

The configuration file for the named daemon is named.conf, located in the /etc directory. It uses a flexible syntax similar to C programs. The format enables easy configuration of selected zones, enabling features such as access control lists and categorized logging. The named.conf file consists of BIND configuration commands with attached blocks within which specific options are listed. Arguments and a block that is delimited with braces follow a configuration command.

Within the block are lines of option and features entries. Each entry is terminated with a semicolon. Comments can use the C, C++, OR SHELL/PERL syntax: enclosing/* */, preceding//, or preceding The following example shows a zone command followed by the zone name and a block options that begin with an opening brace,{.each option entry ends with a semicolon. The entire block ends with a closing brace, also followed by a semicolon.

```
// a caching only name server config
// Zone "." {
Type hint;
File "named.ca";
};
```

The named.conf file is a new feature implemented with BIND version 8.x.the older BIND 4.x version use a file called named.boot.This file is no longer used by version 8.x.the syntax used in these configuration files differ radically. If we upgrade to 8.x, we can use the named-bootconf.pl perl script provided with the BIND software to convert our named.boot file to a named.conf file.

The zone command is used to specify the domains the name server will service. We can enter the keyword zone, followed by the name of the domain placed within double quotes. Period should not be placed at the end of the domain name .if the following example, a Period is within the domain name, but not at the end,"myterk.com"; this differs from the zone file, which requires a period at the end of a complete domain name.

After the zone name, we can specify the class in, which stands for Internet. We can also leave it out, in which case is assumed. Within the zone block, we can place several options. Two essential options are type and file. The type option is used to specify the zone's type. The file option is used to specify the name of the zone file to be used for this zone. We can choose from several types of zones: master, slave, stub, forward, and hint. Master specifies that the zone holds master information and is authorized to act on it.

A master server was called a primary server in the older 4.x BIND configuration. Slave indicates that the zone needs to update its data periodically from a specified master name server .We can use this entry if our name server is operating as a secondary server for another primary (master) DNS server. A stub zone only copies other name server entries, instead of the entire zone. A forward zone directs all queries to name servers specified in a forwarders statement. A hint zone specifies the set of root used by all Internet DNS servers. We can also specify several options that can override any global options set with the options command.

The following example shows a simple zone command for the mytrek.com domain. Its class is Internet," in", and its type is master. The name of its zone file is usually the same as the zone name, in this case,"mytrek.com".

```
Zone "mytrek.com" in {
Type master;
File "mytrek.com";
};
```

DNS BIND Zone types

Type	Description
Master	Primary DNS server
Slave	Slave DNS server controlled by a master DNS server
Hint	Set of root DNS Internet servers.
Forward	Forwards any queries in it to other servers
Stub	Like a slave zone, but holds names of DNS servers.

Other commands, such as acl, server, control, and logging, enable us to configure different features for our name server. The server statement defines the characteristics to be associated with a remote name server, such as the transfer method and key ID for transaction security. The control statement defines a special control associates an authentication method with a particular method with a particular name server. The logging statement is used to configure logging options for the name server, such as the maximum size of the log file and a severity level for messages.

To control access by other hosts, we can use the acl command. Allow and deny options with access control hosts lists enable us to deny or allow access by specified hosts to the name server. The following are the bind commands

Common BIND Configuration Statements and Options

Statement	Description
Include	Includes a file
Key	Specifies key information for use in authentication and authorization

Options	Global server configuration options and defaults for other statements
Server	Sets certain configuration options for the specified server basis
Zone	Defines zone
Type	Specifies a zone type
File	Specifies the zone file for the zone
Directory	Specifies a directory for zone files
Forwarders	Lists hosts for DNS servers where requests are to be forwarded
Masters	Lists hosts for DNS master servers for a slave server
Allow-transfer	Specifies which hosts are allowed to receive zone transfers
Allow-query	Specifies hosts that are allowed make queries
Notify	Allows master servers to notify their slave servers when the master zone data changes and updates are needed

The options statement defines global options and can be used only once in the configuration file. An extensive number of options cover such components as forwarding, name checking, directory path names, access control, and zone transfers, among others. A complete listing can be found in the BIND documentation. A critically important option found in most configuration files is the directory option, which holds the location of the name server's zone and cache files on our system.

The following example is taken from the Red Hat /etc/named.conf file. This example specifies the zone files are located in the /var/named directory. In this directory, we can find our zone files, including those used for our local system.

```
Options
{
directory "/var/named";
forwarders {
        192.168.1.34;
        192.168.1.47;
              };
 };
```

Another commonly used global option is the forwarders option. With the forwarders option, we can list several DNS servers to which queries can be forwarded if they cannot be resolved by the local DNS server. This is helpful for local networks that may need to use a DNS server connected to the Internet. The forwarders option can also be placed in forward zone entries.

With the notify option turned on, the master zone DNS servers send messages to any slave DNS servers whenever their configuration has changed. The slave servers can then perform zone transfers in which they download the changed configuration files. Slave servers always use the DNS configuration files copied from their master DNS servers. Notify takes an argument, yes or no, where yes is default. With the no argument, we can have the master server not send out the messages to the slave servers, in effect, preventing any zone transfers.

The following example is a simple named.conf file based on the example provided in the BIND documentation. The file begins with comments using C++ syntax, //. The options command has a directory entry that sets the directory for the zone and cache files to /var/named. Here, we find your zone files, such as named. Local and reverse mapping files, along with the cache file, named.ca.The first zone command (.) define a hint zone specifying the root name servers.

The cache file listing these servers is name.ca.the second zone command defines a zone for the mytrek.com domain. Its type is master, and its zone file is named "mytrek.com". The next zone is used for reverse IP mapping of the previous zone. Its name is made up of a reverse listing of the mytrek.com domain's IP address with the term in-addr-arpa appended. The domain address for mytrek.com is 192.168.1,so the reverse is 1.168.192.The in-addr-arpa domain is a special domain that supports gateway location and Internet address to host mapping. The last zone command defines a reverse mapping zone for the loop back interface, the method used by the system to address itself and enable communication between local users on the system. The zone file used for this local zone is named. Local.

Named.conf

```
options {
directory"/var/named";
};
};
zone "."{
type hint;
file "named.ca";};
zone "mytrek.com"{
type master;
File "mytrek.com";
};
zone "0.0.127.in-addr.arpa"{
Type master;
file "named. Local";
};
```

When BIND is initially installed, it creates a default configuration for what is known as a caching only server. A caching only server copies queries made by users and saves them in a cache, for use later if the queries are repeated. This can save DNS lookup response times. The cache is held in memory and only lasts as long as named runs. The following example is the named.conf file initially installed for a caching only server. Only the local and cache zones are defined.

Named.conf (caching only server)

```
options{
directory "/var/named";
};
// a caching only name server config
zone "." {
type hint;
file "named.ca";
};
zone "0.0.127.in-addr.arpa"{
type master;
file "named.local";};
```

Resource Records

Our name server holds domain name information about the hosts on our network in resource records placed in zone and reverse mapping files. Resource records are used to associate IP address with fully qualified domain names. We need for every computer in the zone that the name server services. A record takes up one line, though we can use parentheses to use several lines for a record, as is usually the case with SOA records. A resource record uses the Standard Resource record Format as shown here

Name [<ttl>] [<class>] <type><rdata> [<comments>]

Here, name is the name for this record. It can be a domain name for a fully qualified domain name. If we only specify the host name, the default domain is appended. If no name entry exists, then the last specific name is used. If the @ symbol is used, the name server's domain name is used.ttl (time to live) is an optional entry that specifies how long the record is to be cached. Class is the class of the record. The class used in most resource record entries in IN, for the Internet. By default, it is the same as that specified for the domain in the named.conf file. Type is the type of the record. Data is the resource record data. The following is an example of a resource record entry. The name is mytrek.com,the class is Internet(IN),the type is a host address record(A),and the data is the IP address 192.168.1.2.

Mytrek.com.   IN   A   192.168.1.2

A different type of resource records exists for different kinds of hosts and name server operations. A, NS, MX, PTR, and CNAME are the types commonly used. A is used for host address records that match domain names with IP addresses. NS is used to reference a name server. MX specifies the host address of the mail server that services this zone. The name server has n\mail messages sent to that host. The PTR type is used for records that point to other resource records and is used for reverse mapping. CNAME is used to identify an alias for a host on our system.

Domain Name System Resource Records

Type	**Description**
A	Host address, maps host name to IP address
NS	Authoritative name server for this zone
CNAME	Canonical name, used to define an alias for the host name
SOA	Start of Authority, starts DNS entries in zone files, specifies name server for domain and other features like server contact and serial number

Firewall

A firewall prevents any direct unauthorized attempts at access. The firewall is used to protect the local network from the external network such as Internet. The firewall cannot prevent the internal threats. Packet filtering is simply the process of deciding whether a packet received by the firewall host should be passed on into the local network. The common types of firewall are

- Packet Filtering Firewall
- Application level gateway
- Circuit level gateway

Packet Filtering Firewall

A packet filtering firewall applies a set of rules to each incoming IP packet and then forward or discards the packet. The router is typically configured to filter packets going in both directions (from and to the internal network). Filtering rules are based on information contained in a network packet, such as Source IP address, Destination IP address, Source and Destination transport level address, IP Protocol field, interface etc.

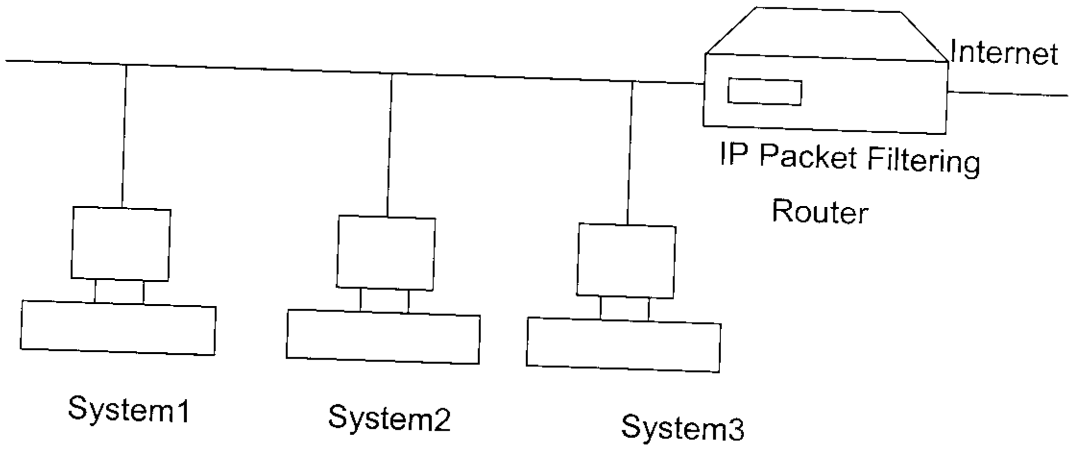
Application level gateway

An application level gateway, also called a proxy server, acts as a relay of application level traffic. The user contacts the gateway using a TCP/IP application, such as Telnet or FTP.

Circuit level gateway

The third type of firewall is the circuit level gateway. This can be a standalone system or it can be a specialized function performed by an application level gateway for certain applications.

Here we are dealing with packet filtering firewall.



PACKET FILTERING FIREWALL

The packet filtering software checks the source and destination addresses of the packet and sends the packet on, if it's allowed. The tool used to implement packet filtering on firewalls for Linux is IPchains. The various kinds of IP chains rules we have to configure are:

1. Input
2. Forward
3. Output

Input

When a packet is received through an interface, the input chain is used to determine what to do with it. The kernel then uses its routing information to decide where to send it.

Forward

If the Kernel sends the packet to another host, then forward chain is checked.

Output

Before the packet is actually sent, the output chain is checked.

For each of these IPchain rules, a separate table of rules is maintained, any of which might refer to one of the user-defined chains.

A firewall rule specifies criteria for a packet, and a target. If the packet does not match, the next rule in the chain is then examined; if it does match, then the next rule is specified by the value of the target, which can be the name of a user-defined chain, or one of the special values ACCEPT, DENY, REJECT, MASQ, REDIRECT, or RETURN. ACCEPT means to let the packet through.

DENY means to drop the packet on the floor. REJECT means the same as drop, but is more polite and easier to debug, since an ICMP message is sent back to the sender indicating that he packet was dropped. (Note that DENY and REJECT are the same for ICMP packets). MASQ is only legal for the forward and user defined chains, and can only be used when the kernel is compiled with CONFIG_IP_MASQUERADE defined.

With this, packets will be masqueraded as if they originated from the local host. Furthermore, reverse packets will be recognized as such and they will be demasqueraded automatically, bypass-ing the forwarding chain. REDIRECT is only legal for the input and user-defined chains and can only be used when the Linux kernel is compiled with CONFIG_IP_TRANSPARENT_PROXY defined. With this, packets will be redirected to a local socket, even if they were sent to a remote host.

If the specified redirection port is 0, which is the default value, the destination port of a packet will be used as the redirection port. When this target is used, an optional extra argument (the port number) can be supplied. If the end of a user-defined chain is reached, or a rule with target RETURN is matched, then the next rule in the previous (calling) chain is examined. If the end of a built-in chain is reached, or a rule in a built-in chain with target RETURN is matched, the target specified by the chain policy determines the fate of the packet.

IPchains Options

Options	**Function**
-A chain	Append a rule to a chain
-D chain	Delete (flush) matching rule from chain
-D chain rulenum	Delete rule rulenum (1=first) from chain
-I chain [rulenum]	Insert in chain as rulenum (default 1=first)
-R chain rulenum	Replace rule rulenum (1=first) in chain
-L [chain]	List the rules in chain or all chains
-F [chain]	Delete (flush) all rules in chain or all chains
-Z chain	Zero counters in chain or all chains
-C chain	Test this packet on chain
-N chain	Create a new user-defined chain
-X chain	Delete a user-defined chain

The following example accepts messages coming in that are from (source) any host in the 192.168.0.0 network and that are going (destination) anywhere at all (he –d option s left out or could be written as –d 0/0).

ipchains –A input –s 192.168.0.0/24 –j ACCEPT

The following example restricts access to 192.168.1.166,denyng access to all others.

ipchains –A input ! –s 192.168.1.166 –j DENY

Parameters

The following parameters make up a rule specification (as used in the add, delete, replace, append and check com-mands).

-p, --protocol[!] protocol

The protocol of the rule or of the packet to check. The specified protocol can be one of tcp, udp, icmp, or all, or it can be a numeric value, representing one of these protocols or a different one. Also a protocol name from /etc/protocols is allowed. A "!" Argument before the protocol inverts the test. The number zero is equivalent to all. Protocol all will match with all protocols and is taken as default when this option is omitted. All may not be used in combination with the check command.

-s, --source, --src [!] address[/mask] [!] [Port [: port]]

Source specification. Address can be a hostname, a network name, or a plain IP address. The mask can be either a network mask or a plain number, specifying the number of 1's at the left side of the network mask. Thus, a mask of 24 is equivalent to 255.255.255.0. A "!" argument before the address specification inverts the sense of the address. The source may include a port specification or ICMP type. This can either be a service name, a port number, a numeric ICMP type, or one of the ICMP type names shown by the command

ipchains -h icmp

Note that many of these ICMP names refer to both a type and code, meaning that an ICMP code after the -d flag is illegal. In the rest of this paragraph, a port means either a port specification or an ICMP type. An inclusive range can also be specified, using the format port: port. If the first port is omitted, "0" is assumed; if the last is omitted, "65535" is assumed. Ports may only be specified in combination with the tcp, udp, or icmp protocols. A "!" before the port specification inverts the sense. When the check command is specified, exactly one port is required, and if the -f (fragment) flag is specified, no ports are allowed.

--source-port [!] [port[:port]]

This allows separate specification of the source port or port range. The flag --sport is an alias for this option.

-d, --destination, --dst [!] address[/mask] [!] [port[:port]]

The -s (source) flag for a detailed description of Destination specification. For ICMP, which does not have ports, a "destination port" refers to the numeric ICMP code.

--destination-port [!] [port[:port]]

This allows separate specification of the ports. The flag --dport is an alias for this option.

--icmp-type [!] typename

This allows specification of the ICMP type (use the -h icmp option to see valid ICMP type names). This is often more convenient than appending it to the destination specification.

-j, --jump target

This specifies the target of the rule that is what to if the packet matches it. The target can be a user-defined chain (not the one this rule is in) or one of the special targets, which decide the fate of the packet immediately. If this option is omitted in a rule, then matching the rule will have no effect on the packet's fate, but the counters on the rule will be incremented.

-i, --interface [!] name

Optional name of an interface via which a packet is received (for packets entering the input chain), or via which is packet is going to be sent (for pack-ets entering the forward or output chains). When this option is omitted, the empty string is assumed, which has a special meaning and will match with any interface name. When the "!" Argument is used before the interface name, the sense is inverted. If the interface name ends in a "+", then any interface which begins with this name will match.

[!] -f, --fragment

This means that the rule only refers to second and further fragments of fragmented packets. Since there is no way to tell the source or destination ports of such a packet (or ICMP type), such a packet will not match any rules which specify them. When the "!" argument precedes the "-f" flag, the sense is inverted.

Test Case – Service Selection

Test ID	Description	Procedure and Expected Results	Results
S_01	Administrator login	**Procedure** The administrator is asked to enter his password **Expected Result** A screen for choosing the type of service to be configured is displayed If there is any error in the password his login is denied.	Passed
S_02	Service selection	**Procedure** The administrator selects the service he wants to configure. **Expected Result** The administrator is displayed with the corresponding configuration page.	Passed

Test Case – Proxy

Test ID	Description	Procedure and Expected Results	Results
P_01	Url restriction/Permission	**Procedure** The administrator enters the site names that he wants to restrict/permit. **Expected Result** The corresponding acl for restricting/permitting the search word are is written in the squid.conf file. If no site is entered in the text box an error message is given.	Passed
P_02	Search word restriction	**Procedure** The administrator enters the search words that he wants to restrict/permit. **Expected Result** The corresponding acl for restricting/permitting the sites are to be written in the squid.conf file. If no word is entered in the text box an error message is given	Passed

Test ID	Description	Procedure and Expected Results	Results
P_03	Session control	**Procedure** The administrator enters the number of http connections that he wants to permit at a time **Expected Result** The corresponding acl for controlling the http connection is written in the squid.conf file. If number is not entered in the text box an error message is given.	Passed
P_04	IP filtering	**Procedure** The administrator enters the client IP address that he wants to restrict/permit. **Expected Result** The corresponding acl for restricting/permitting the client is written in the in the squid.conf file. If no address is entered in the text box an error message is given	Passed

Test Case - DNS

Test ID	Description	Procedure and Expected Results	Results
D_01	Adding a zone	**Procedure** The administrator enters the zone name and the corresponding resource records values. **Expected Result** The zone gets added and responds to the 'dig' command. Error message is displayed if any wrong entry is given.	Passed
D_02	View and change an existing zone	**Procedure** The administrator can view the existing zones in the DNS and can make changes to the resource record values if needed. **Expected Result** The details of resource records are provided and changes made are updated accordingly.	Passed
D_03	Delete a zone	**Procedure** The administrator deletes a zone if required. **Expected Result** The zone is removed from the domain name server.	Passed

Test Case - Firewall

Test ID	Description	Procedure and Expected Results	Results
F_01	Denying a protocol	**Procedure** The administrator selects the protocol to be denied and the source and destination to which the rule should be applied. **Expected Result** Access is denied for the specified protocol under the specified criteria.	Passed
F_02	Redirecting to a port	**Procedure** The administrator selects the' from port' and the 'to port' for the redirection to take place. **Expected Result** When the packets arrive at the 'from port' it gets automatically redirected to the 'to port'	Passed

6. FUTURE ENHANCEMENTS

The following are the future enhancements that are possible in our project:

- Inclusion of more options for the configuration of proxy such as user authentication and some other facilities available in squid.
- Perform content based filtering in the firewall configuration.
- Include mail exchanger record in the DNS configuration.
- Make our project a platform independent one.

7. CONCLUSION

The anticipated project is completed successfully. Thus interface for configuring the proxy, firewall and the DNS is designed using PHP. The interface is very user friendly and the administrator or the person who configures need not have a thorough knowledge about the configuration. The interface is capable of successfully configuring the three-security service. The three configurations are presented as a single interface, which is a central advantage of our project.

8. REFERENCES

Bibliography:

1.Elizabeth D.Zwicky, Simon Cooper And D.Brent Chapman," *Building Internet Firewalls*", Shroff publishers and Distributors Pvt Ltd, 2nd Edition.

2.Ashok Appu "*PHP A Beginner's Guide*", Wiley Publishing Inc.1st Edition.

3.Richard Petersen "*The Complete Reference Linux*", Tata McGraw Hill Publishing Limited.4th Edition.

4.Hunt, Craig" *Linux DNS Server Administration*", BPB Publication, 2nd Edition.

5.Albitz, Paul, Liu, Cricket,"*DNS and BIND*", Shroff publishers and Distributors Pvt Ltd, 3rd Edition.

6.Concalves, Marcus, "*Firewall: A Complete Guide*", Tata McGraw Hill Publishing Limited, 3rd Edition.

7.Rasmus Lerdorf and Kevin Tatroe,"*Programming PHP*", Shroff publishers and Distributors Pvt Ltd, 1st Edition

Web sites

www.squid-cache.org

www.squidguard.org

www.dns.net

www.fwbuilder.org/pages/documents

www.php.net

9. APPENDIX

9.1 Sample source code

Module 1 – Proxy

```
<html>
<head>
<title>
PASSWD</title></head>
<body background="a.gif">
<font bgcolor="#e43117" size="15"><br><br>
<center><B>USER AUTHENTICATION</B></CENTER></font>
<form name="form1" action="passwd.php" method="POST"><br><br><br><br>
<table width="350" border="5" align="center" cellpadding="5" cellspacing="5">
<tr>
<td bgcolor="#dddddd"><div align="center"><b>ENTER THE PASSWORD TO
VIEW THE PFD PAGE</b></div>
</td></tr>
<tr bgcolor="#dddddc">
<td height="10">Password
<input type="password" name="password"><BR><BR>
<center><input type="submit" name="submit" value="LOGIN"></center>
</td></tr>
</table></form></body>
<html>
<head>
<title>
authentication</title>
</head>
<body background="forest2.jpg">
```

```
<?php
if(empty($password))
{
die("<h1><font color=pink><center>no passwd
mentioned</center></font></h1>");
}
if(!($password=="psgs"))
{
die("<h1><font color=pink><center>password not valid</center></font></h1>");
}
?>
<br><br><p>
<center><font size=30 color=dddddc>You are accepted as a valid
user.</font></center>
</p><br><br><br><p>
<font size=15 color=dddddc>Click to view the <a href="button.html"><i>PFD
Joint Effort Services setup</i></a></font>
</p><br> <form action="button.html" method="GET">
<input type="submit" value="NEXT">
</form></body></html>
<html><body><?php
$f="/etc/squid/squid.conf";
$file="/etc/sites";
if(! `grep -n -i -w "^[.[:space:]]acl site" /etc/squid/squid.conf`)
{
$f1=fopen($f,"a+") or die("could not open squid file");
fwrite($f1,"\n acl site url_regex -i \"/etc/sites\" \n http_access     deny site\n");
fclose($f1);
}
$tem=ereg_replace('[[:space:]]',',',$words);
$tem=ereg_replace('[,]+',',',$tem);
```

```
$temp=explode(',',$tem);
print "<br>"; $j=count($temp);
$fp=fopen($file,"a+") or die("could not open word file");
for($i=0;$i<$j;$i++)
{ if(! `grep -i -w  $temp[$i] /etc/sites`)
{      fwrite($fp,"$temp[$i] \n");
$flag=1;   }
if ($flag==1)
{ print "<br>";
echo "$temp[$i] successfully restricted";
}
else
{print "<br>";
echo "$temp[$i] already restricted";
}
}
?>
</body></html></html>
<html>
<body>  <?php   $f="/etc/squid/squid.conf";
$file="/etc/sites";
if(! `grep -n -i -w "^[.[:space:]]acl site" /etc/squid/squid.conf`)
{
$f1=fopen($f,"a+") or die("could not open squid file");
fwrite($f1,"\n acl site url_regex -i \"/etc/sites\" \n http_access deny site\n");
fclose($f1);
}$tem1=$pwords;
print "<br>";
$tem1=ereg_replace('[[:space:]]',',',$pwords);
$tem1=ereg_replace('[,]+',',',$tem1);
$temp1=explode(',',$tem1);
```

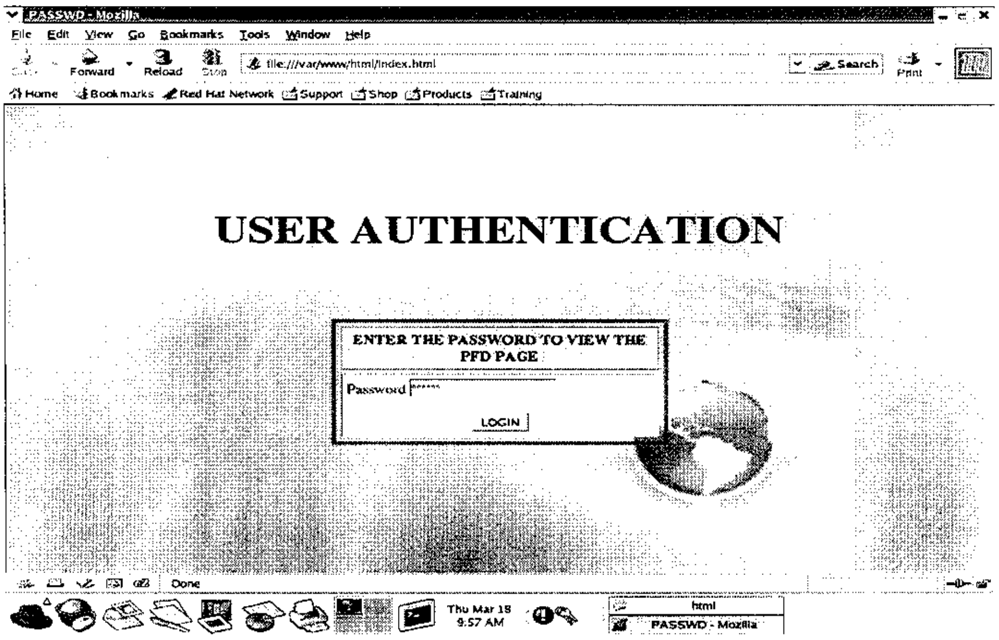
```
print "<br>"; $j1=count($temp1);
$flag1=0;
for($i1=0;$i1<$j1;$i1++)
{
if ( `grep $temp1[$i1] /etc/sites `)
{ $flag1=1;
`grep -v $temp1[$i1] /etc/sites > /etc/sites1`;
`cat /etc/sites1 > /etc/sites`;
print "<br>";
echo " $temp1[$i1] successfully permited";
}
else
{
print "<br>";
echo " $temp1[$i1] was not restricted";
}
}
?>
</body></html>
<html><body><?php
$f="/etc/squid/squid.conf";
$file="/etc/search";
if(! `grep -n -i -w "^[.[:space:]]acl sword" /etc/squid/squid.conf`)
{
$f1=fopen($f,"a+") or die("could not open squid file");
        fwrite($f1,"\n acl sword urlpath_regex -i \"/etc/search\" \n http_access deny
        sword\n");     fclose($f1);
}
$tem=ereg_replace('[[:space:]]',',',$words);
$tem=ereg_replace('[,]+',',',$tem);
$temp=explode(',',$tem);  print "<br>";$j=count($temp);
```

```
$fp=fopen($file,"a+") or die("could not open word file");
for($i=0;$i<$j;$i++)
{ if(! `grep -i -w  $temp[$i] /etc/sites`)
{      fwrite($fp,"$temp[$i] \n");
$flag=1;
}
if ($flag==1)
{ print "<br>";
echo "$temp[$i] successfully restricted";
}
else
{
print "<br>";
echo "$temp[$i] already restricted";
}
}
?></body></html>
<html><head><title>
IP FILTERING
</title></head><body>
<b><center><font size="30"> IP FILTERING </center></b></font>
<br><br><br>
<center><font size="20"><b><i>SELECT THE WAY U WANT TO RESTRICT
THE IP PACKETS
</i></font></b></center><br><br><br><center><font size="10"><i>
<a href="cliip.html" >client's ip address</a><br><br>
<a href="rng.html" >range of addresses</a><br><br>
<a href="host.html" >URL host's ip address</a><br><br>
</body></html>
<html><body><?php
$f="/etc/squid/squid.conf";
```

```
$file="/etc/client";
$fp=fopen($file,"a+") or die("could not open word file");
if(! `grep -n -i -w "^[.[:space:]]acl cliip" /etc/squid/squid.conf`)
{ $f1=fopen($f,"a+") or die("could not open squid file");
$arr=explode(':',`grep -b "^#[.[:space:]]INSERT" /etc/squid/squid.conf`);
echo $arr[0];
if(fseek($f1,$arr[0],SEEK_SET)==-1)
{ echo "error"; exit(0); }
$t2=fgetc($f1);
while(1)
{ $t1=$t2;
$t2=fgetc($f1);
if($t1=="\n" && $t2!="#")
{ fseek($f1,-1,SEEK_CUR);
break;
}
}fwrite($f1,"\n\n");
fwrite($f1,"\n acl client src  \"/etc/site\" \n http_access deny client\n");
fclose($f1);
}
echo "$words";
fwrite($fp,"$words");
fclose($fp);
?></body></html>
<html><body><?php
$f="/etc/squid/squid.conf";
$file="/etc/range";
$fp=fopen($file,"a+") or die("could not open word file");
if(! `grep -n -i -w "^[.[:space:]]acl rng" /etc/squid/squid.conf`)
{ $f1=fopen($f,"a+") or die("could not open squid file");
fwrite($f1,"\n acl rng src \"/etc/range\" \n http_access deny rng\n");
fclose($f1);
}
echo "$ip2";
fwrite($fp,"$ip2\n");
fclose($fp);
?>
</body></html>
<html><body><?php
$f="/etc/squid/squid.conf";
$file="/etc/host";
$fp=fopen($file,"a+") or die("could not open word file");
if(! `grep -n -i -w "^[.[:space:]]acl hos" /etc/squid/squid.conf`)
{ $f1=fopen($f,"a+") or die("could not open squid file");
fwrite($f1,"\n acl hos dst \"/etc/host\" \n http_access deny hos\n");
fclose($f1);
}
echo "$ip3";
fwrite($fp,"$ip3\n");
fclose($fp);
?>
</body>
</html>
```

9.2 SNAPSHOTS

PASSWORD VALIDATION


PASSWD - Mozilla
File Edit View Go Bookmarks Tools Window Help
Forward Reload Stop
file:///var/www/html/index.html
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
Done
Thu Mar 18 9:57 AM
html
PASSWD - Mozilla

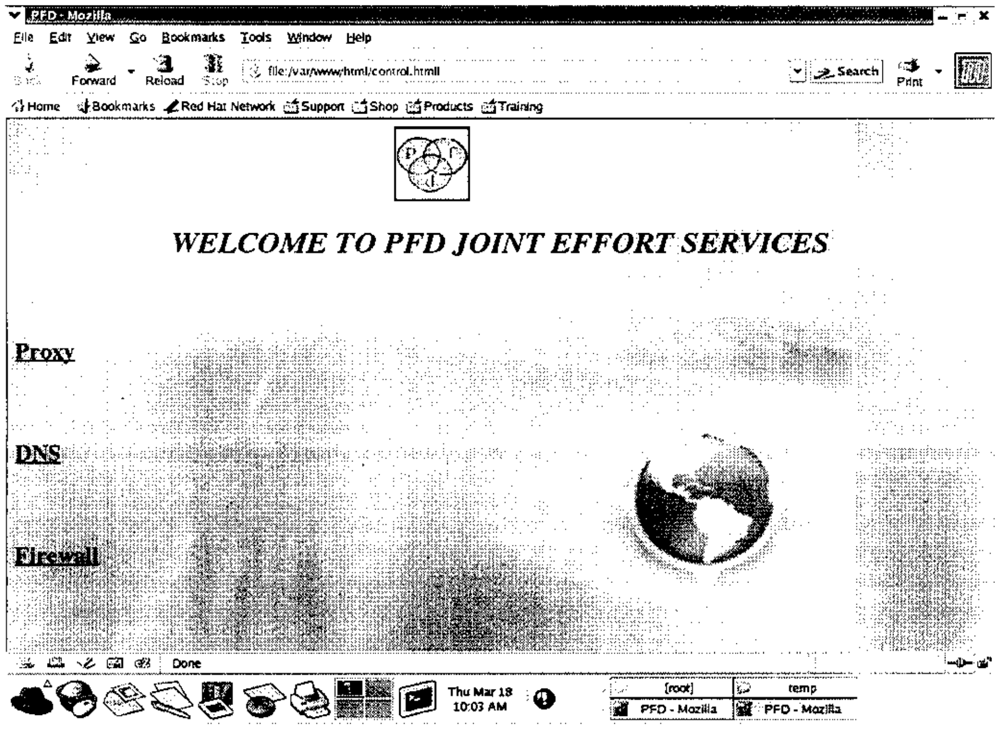
USER AUTHENTICATION

ENTER THE PASSWORD TO VIEW THE PFD PAGE

Password ******

LOGIN

PFD INDEX PAGE


PFD - Mozilla
File Edit View Go Bookmarks Tools Window Help
Forward Reload Stop
file:/var/www/html/control.html
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
Done
Thu Mar 18 10:03 AM
[root]
temp
PFD - Mozilla
PFD - Mozilla

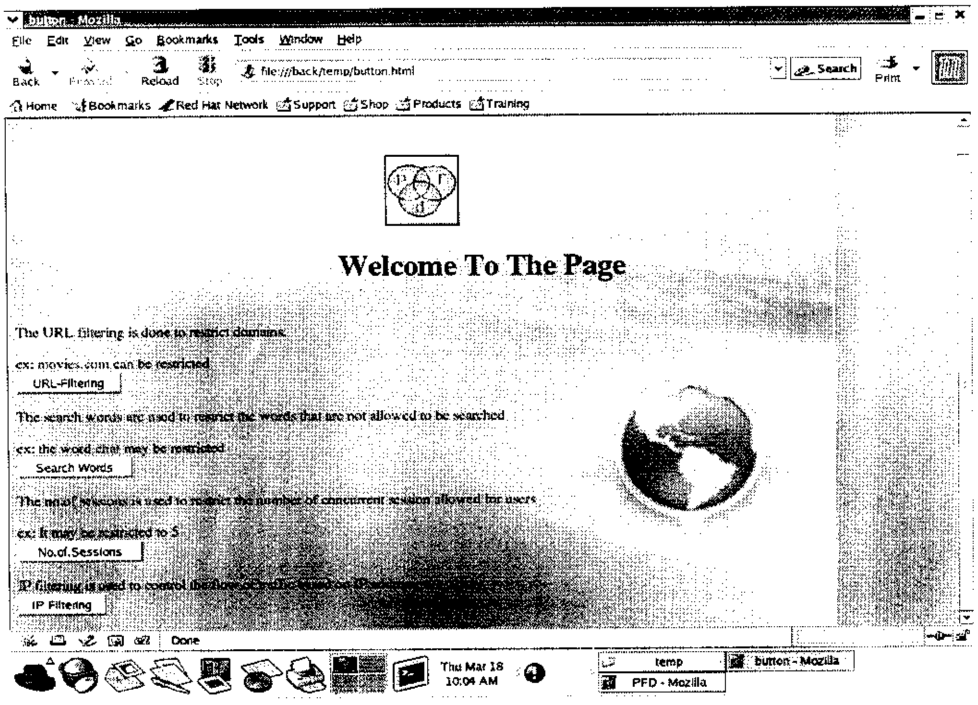
WELCOME TO PFD JOINT EFFORT SERVICES

Proxy

DNS

Firewall

PROXY – OPTIONS



Welcome To The Page

The URL filtering is done to restrict domains.

ex: movies.com can be restricted

URL-Filtering

The search words are used to restrict the words that are not allowed to be searched

ex: the word chat may be restricted

Search Words

The no.of sessions is used to restrict the number of concurrent session allowed for users

ex: It may be restricted to 5

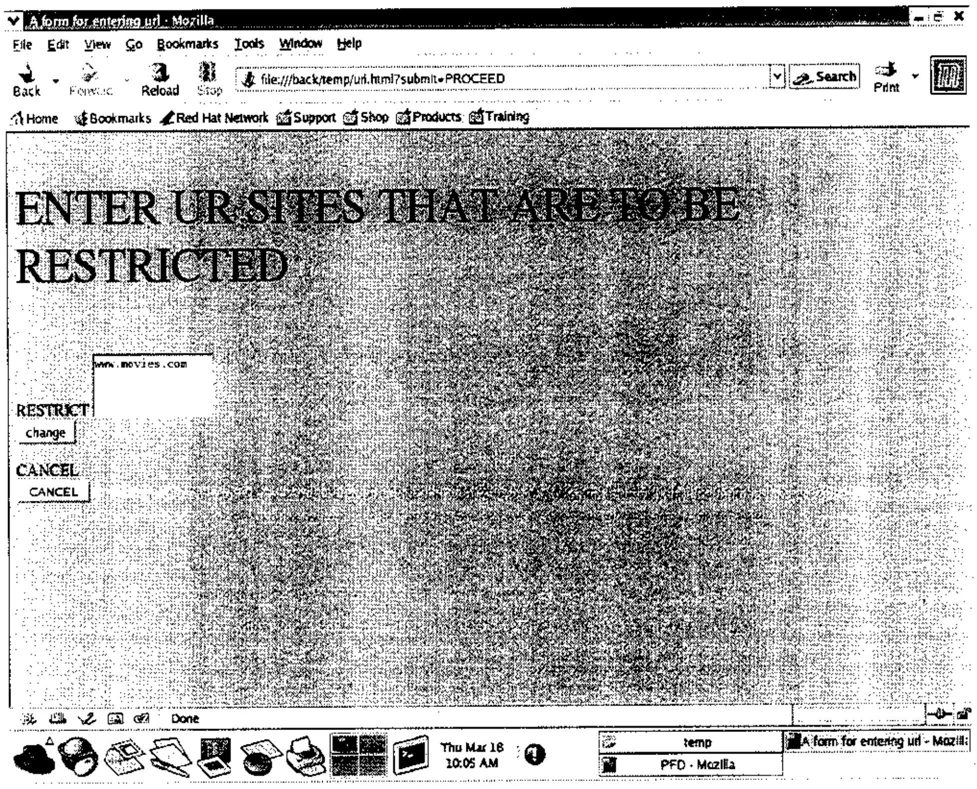
No.of.Sessions

IP filtering is used to control the flow of traffic based on IP [illegible]

IP Filtering

button - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
file:///back/temp/button.html
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
Done
Thu Mar 18 10:04 AM
temp
button - Mozilla
PFD - Mozilla

URL FILTERING



ENTER UR SITES THAT ARE TO BE RESTRICTED

www.movies.com

RESTRICT

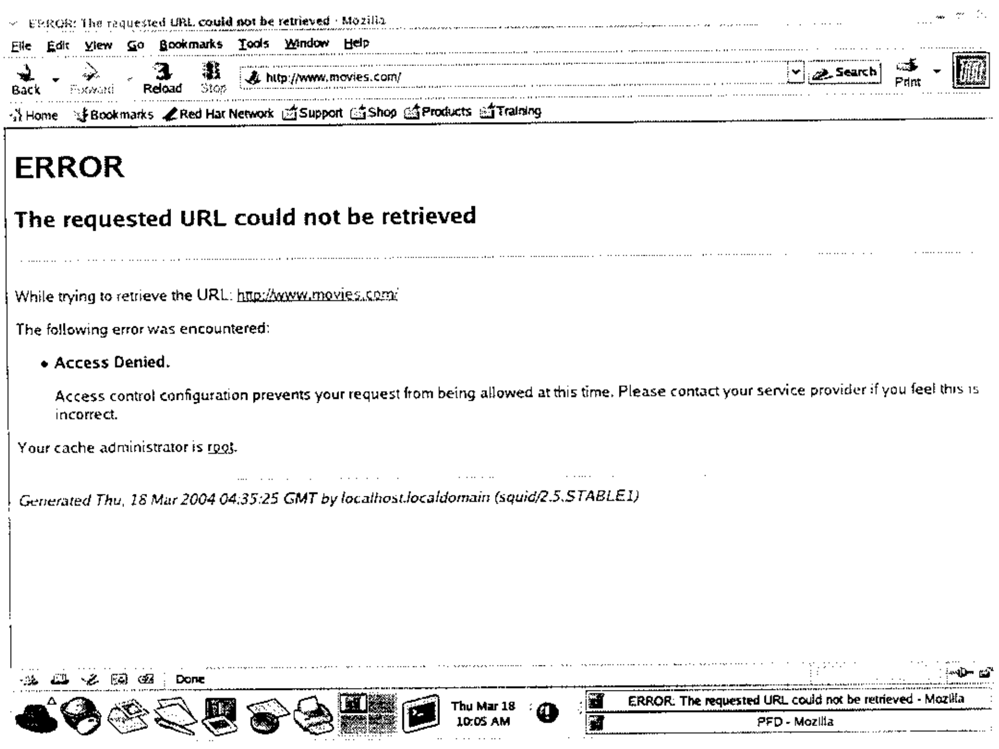
change

CANCEL

CANCEL

A form for entering url - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
file:///back/temp/url.html?submit=PROCEED
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
Done
Thu Mar 18 10:05 AM
temp
A form for entering url - Mozill
PFD - Mozilla

URL FILTERING – OUTPUT


ERROR: The requested URL could not be retrieved - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://www.movies.com/
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
Done
Thu Mar 18 10:05 AM
ERROR: The requested URL could not be retrieved - Mozilla
PFD - Mozilla

ERROR

The requested URL could not be retrieved

While trying to retrieve the URL: http://www.movies.com/

The following error was encountered:

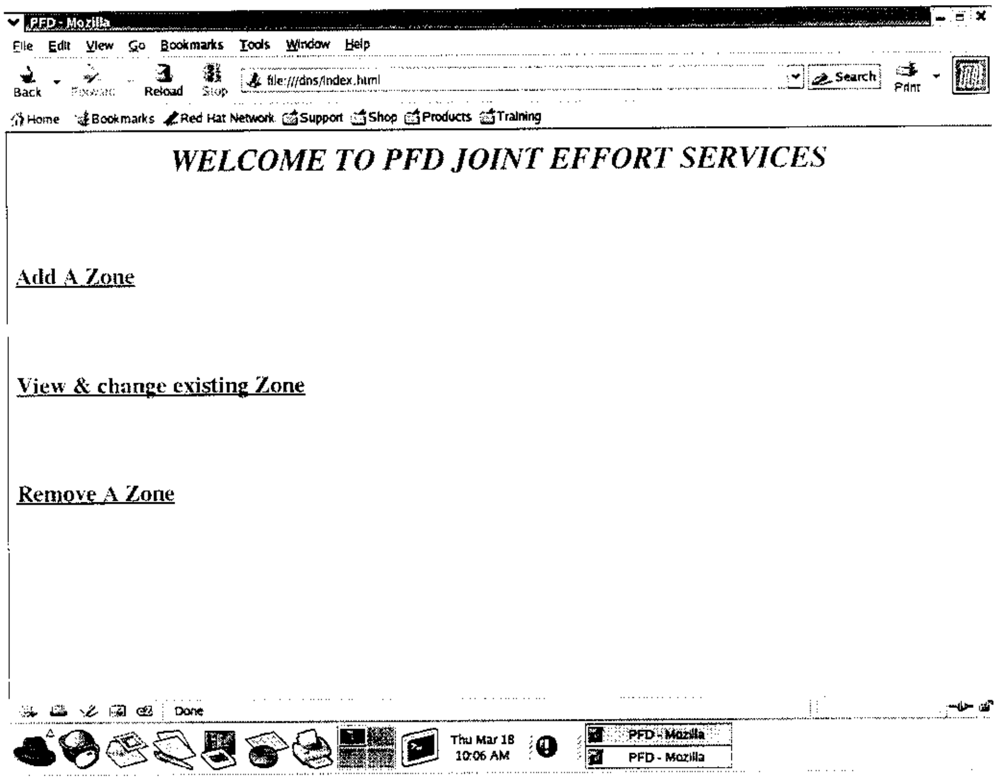
- Access Denied.

  Access control configuration prevents your request from being allowed at this time. Please contact your service provider if you feel this is incorrect.

Your cache administrator is root.

Generated Thu, 18 Mar 2004 04:35:25 GMT by localhost.localdomain (squid/2.5.STABLE1)

DNS OPTIONS



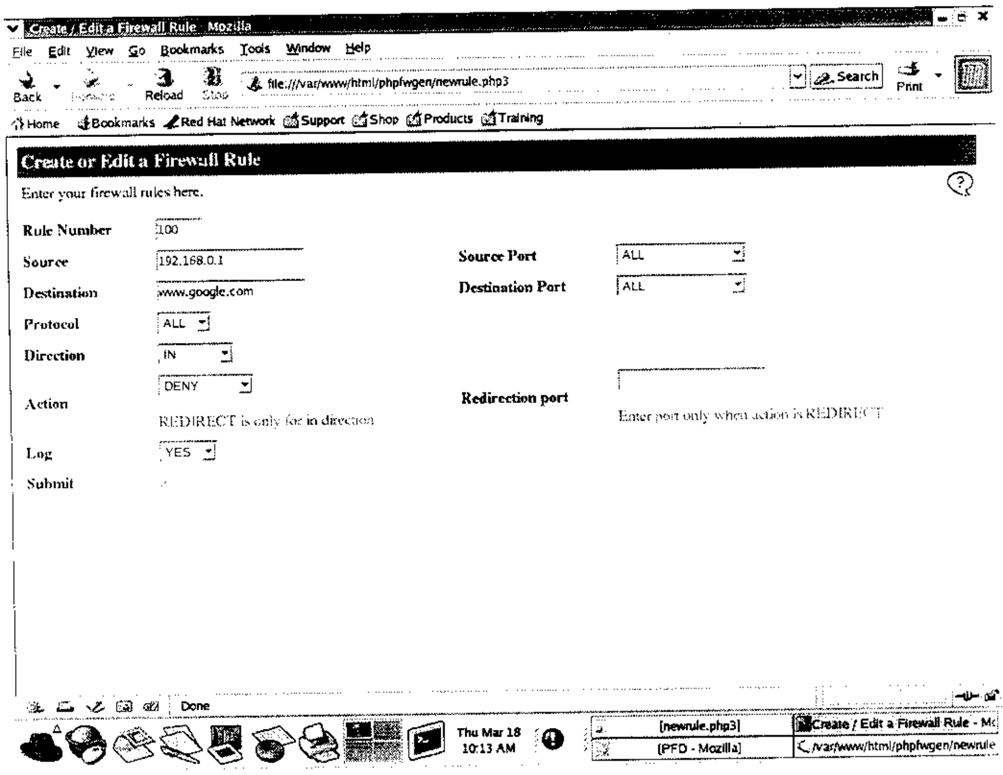
WELCOME TO PFD JOINT EFFORT SERVICES

Add A Zone

View & change existing Zone

Remove A Zone

FIREWALL – EDIT RULES

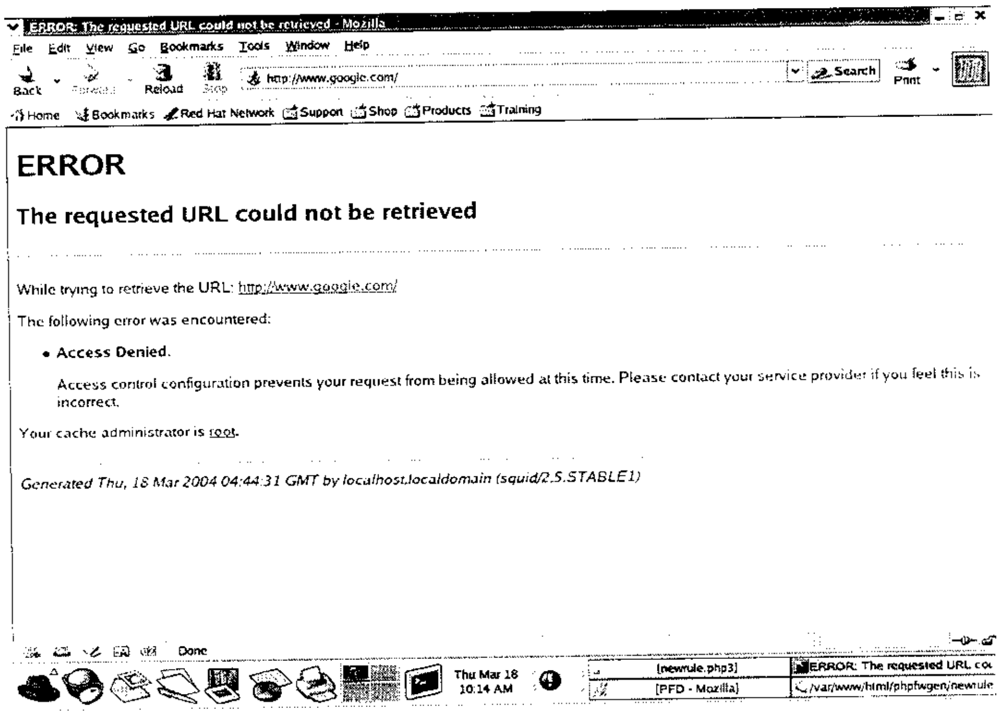

Create / Edit a Firewall Rule - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
file:///var/www/html/phpfwgen/newrule.php3
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
Done
Thu Mar 18 10:13 AM
[newrule.php3]
[PFD - Mozilla]
Create / Edit a Firewall Rule - Mc
/var/www/html/phpfwgen/newrule

Create or Edit a Firewall Rule

Enter your firewall rules here.

Rule Number	100		
Source	192.168.0.1	Source Port	ALL
Destination	www.google.com	Destination Port	ALL
Protocol	ALL		
Direction	IN		
Action	DENY REDIRECT is only for in direction	Redirection port	Enter port only when action is REDIRECT
Log	YES		
Submit			

FIRE WALL – OUTPUT


ERROR: The requested URL could not be retrieved - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://www.google.com/
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
Done
Thu Mar 18 10:14 AM
[newrule.php3]
[PFD - Mozilla]
ERROR: The requested URL co
/var/www/html/phpfwgen/newrule

ERROR

The requested URL could not be retrieved

While trying to retrieve the URL: http://www.google.com/

The following error was encountered:

- **Access Denied.**

  Access control configuration prevents your request from being allowed at this time. Please contact your service provider if you feel this is incorrect.

Your cache administrator is root.

Generated Thu, 18 Mar 2004 04:44:31 GMT by localhost.localdomain (squid/2.5.STABLE1)