

Stateful Multilayer Inspection Firewall



Under the supervision of

Prof. Dr.S.P. Ghrera

By

Ampin Gupta (101337)

**DEPARTMENT OF COMPUTER SCIENCE AND
INFORMATION TECHNOLOGY**

**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY -
WAKNAGHAT**

Part-1 Packet Filtering Firewall

- (a) Learning to filter packets.**
- (b) Creating a packet filter firewall.**
- (c) Write packet filtering rules.**
- (d) Testing and debugging.**

Part-2 Circuit and Application Level Gateway Firewall

- (a) Creating circuit level gateway filtering firewall.**
- (b) Creating application level gateway filtering firewall.**
- (c) Integration.**
- (d) Test run.**
- (e) Real-time evaluation of Stateful Multilayer Inspection Firewall .**

CERTIFICATE

This is to certify that the work entitled, "**Stateful Multilayer Inspection Firewall - part1: Packet filtering firewall**" submitted by: Ampin Gupta - 101337, in partial fulfilment for the award of degree of Bachelor of Technology in **Computer Science Engineering** of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute before the award of this or any other degree or diploma.

Date:

Prof. Dr. Satya Prakash Ghrera,
Professor, Brig (Retd.) and Head,
Department of Computer Science and Information Technology

Jaypee University of Information Technology
Waknaghat

ACKNOWLEDGEMENT

I would like to place on record our deep sense of gratitude to Prof. **Dr. S.P. Ghrrera**, HOD-Dept. of Computer Science and Information Technology, JUIT, Wajnaghat, India for his generous guidance, help and useful suggestions. I appreciate his immense help and his feedbacks which helped us in developing the project so successfully.

I would also like to thank him for providing us with valuable help in learning about Networks and the various firewall mechanisms. His insightful thinking and constructive suggestions have helped us improve the quality of this project work.

Ampin Gupta (101337)

PROBLEM STATEMENT

To control the incoming and outgoing network traffic by analyzing the data packets and be able to determine whether it should be allowed through or not, based on a predetermined rule set.

The project necessitated a thorough study and implementation of Firewall Security Mechanisms. As such the main focus of the project has been to deal with the problem of *security over networks* where many computers are sharing data.

In internet and similar networks, connecting an organisation to the internet provides a two-way flow of traffic. This clearly is undesirable in many organisations, as proprietary information is often displayed deeply within a corporate intranet. In order to provide some level of separation between an organisation's intranet, and the internet, firewalls have been employed.

Various firewalls have been designed and are being designed for the purpose of network security where the filtering rules are defined by the OS itself and are complex to modify.

Through this project we seek to develop a new and better packet filtering firewall that is more efficient in terms of time and cost and is also user-friendly and covers a software based implementation, thus, making it easy to configure.

OBJECTIVE OF THE PROJECT

To develop a packet filtering firewall as a part of the Stateful Multi-Layer Filtering Firewall, that allows or block traffic as per the specified rule set.

Description:

We shall be building a packet filtering firewall, by writing out packet filtering rules. To implement this, we shall be using a driver to invoke a system service. This system service will then run the firewall application. Also, the firewall shall run in the background once it has started.

The firewall application shall have functionalities to add rule(s), modify a rule, delete rule. Based on the rule defined, the packet passing the firewall shall either be blocked or be allowed to pass.

TABLE OF CONTENTS

1. Introduction.....	1
1.1 What is a firewall.....	1
1.2 Tasks of a firewall.....	1
1.3 Packet filtering-.....	1
1.3.1 Stateless packet filtering.....	1
1.3.2 Stateful Packet Filtering.....	2
1.4 What Should Be Inspected In A Packet Header.....	2
1.5 Advantages and disadvantages.....	2
2. Literature Review.....	3
3. Project Background.....	9
3.1 Packet filtering	9
3.2 Functions of a Packet Filtering Firewall.....	10
3.3 Types of filtering mechanisms:.....	10
3.3.1 Packet filters.....	11
3.3.2 Circuit level gateways.....	11
3.3.3Application level gateways.....	11
3.3.4 Stateful multilayer inspection firewalls.....	12

4. Project Design.....	13
4.1 Model used.....	13
4.2 Input/ output.....	14
4.3 Flow Chart.....	15
4.4 Use Case.....	16
5. Implementation.....	16
5.1 Working.....	17
5.2 Description of classes used.....	18
6. Result and discussions.....	31
6.1 Conclusion	41
6.2 Implications for future research.....	41
7. Bibliography.....	42

INTRODUCTION

Def.: A *firewall* is any security system protecting the boundary of an internal network.

Tasks of a firewall:

- access control based on sender or receiver address
- access control based on services requested
- hiding the internal network, e.g. topology, addresses, etc.
- virus checking on incoming files
- authentication based on the source of traffic
- logging of Internet activities

Placement of firewall components:

- (1) packet filter
- (2) circuit-level proxy
- (3) application-level proxy

Packet Filtering:

Packet filtering is a network security mechanism that works by controlling what data can flow to and from a network. The data that flows is in the form of packets.

Packet filtering firewall allows only those packets to pass, which are allowed as per your firewall policy. Each packet passing through is inspected and then the firewall decides to pass it or not. The packet filtering can be divided into two parts:

- I. Stateless packet filtering.
- II. Statefull packet filtering.

Stateless Packet Filtering

If the information about the passing packets is not remembered by the firewall, then this type of filtering is called stateless packet filtering. This type of firewalls are not smart enough and can be fooled very easily by the hackers. These are especially dangerous for UDP type of data packets. The reason is that, the allow/deny decisions are taken on packet by packet basis and these are not related to the previous allowed/denied packets.

Stateful Packet Filtering

If the firewall remembers the information about the previously passed packets, then that type of filtering is stateful packet filtering. These can be termed as smart firewalls. This type of filtering is also known as Dynamic packet filtering.

What Should Be Inspected In A Packet Header?

- 1. Source IP address of the packet.** This is necessary because IP spoofers might have changed the source IP address to reflect the origin of packet from somewhere else, rather than reflecting the original source.
- 2. Destination IP Address.** The firewall rules should check for IP address rather than DNS names. This prevents abuse of DNS servers.
- 3. IP Protocol ID.**
- 4. TCP/UDP port number.**
- 5. ICMP message type.**
- 6. Fragmentation flags.**
- 7. IP Options settings.**

Advantages Of Packet Filtering Firewall :

1. Because not a lot of data is analyzed or logged, they use very little CPU resources and create less latency in a network. They tend to be more transparent in that the rules are configured by the network administrator for the whole network so the individual user doesn't have to face the rather complicated task of firewall rule sets.
2. It is cost effective to simply configure routers that are already a part of the network to do additional duty as firewalls.

Disadvantages:

1. Packet Filtering Firewalls can work only on the Network Layer and these Firewalls do not support complex rule based models. And it's also vulnerable to Spoofing in some cases.

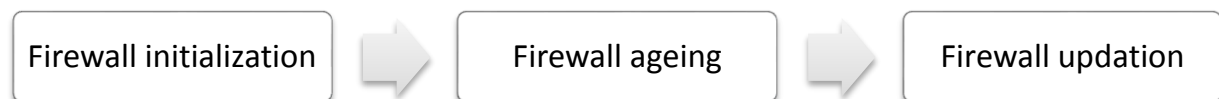
LITERATURE REVIEW

[1]V.K. Solanki, K.P. Singh, M. Venkatesan, S. Raghuwanshi, , "Firewalls Policies Enhancement Strategies Towards Securing Network", Dept. of CSE, Anna Univ., Chennai, India , in Proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT 2013), 11-12 April 2013.

Brief Summary:

This paper is aimed at discussing about the enhancement of firewalls policies in the network. The Firewalls can be improved in multiple ways but in practical scenario, there are many conflicts like first is not going for dynamic updating of firewalls policies and second the policies conflict with LAN and WAN.

Three stage policies approach:



Firewall initialization- the system security team member will first study the nature of network and initialize the rule.

Firewall ageing - to identify the usage of policy.

Firewall updation - the final rule deployment is done to make it effective.

Advantage: with this three stages approach to make firewall more efficient is easy for network security team member to protect their network from outside as well as inside unauthorized and unwanted access.

Disadvantage: We have realized that the approach is not very useful in case if the policies involved in firewalls are misuses by internal employees and second the planning of policies if not done properly then it's dicey to say that using above approach make firewalls more efficient and more optimize towards securing the main aim of network security.

With the help of three stage designed approached we have realized that *if its run by cost-on-cost basis than the efficiency of the firewall could be certainly improved.* We have found the novel idea to increase the effectiveness of firewalls through optimal policies change usage in network using three stages policy approach.

[2] Tugkan Tuglular, Fevzi Belli, "*Protocol-Based Testing of Firewalls*", Department of Computer Science, Dept. of Comput. Eng., Izmir Inst. of Technol., Izmir, Turkey , 2009 Fourth South-East European Workshop on Formal Methods.

Brief Summary:

A firewall should be tested rigorously with respect to its implemented network protocols and security policy specification. Abstract test cases are generated by mutating event sequence graph model of chosen network protocol and filled with values from policy specification by using equivalence partitioning and boundary value analysis. A case study is presented to validate the presented approach.

There are three general approaches to firewall testing: penetration testing, testing of the firewall rules, and testing of the firewall implementation.

Penetration testing is performed to check the firewall for potential breaches of security that can be exploited.

The firewall penetration testing is structured in the following four steps :

- indirect information collection
- direct information collection
- attack from the outside
- attack from the inside.

This type of testing targets specific known vulnerabilities of firewalls determined through information collection.

Testing of the firewall rules verifies whether the security policy is correctly enforced by a sequence of firewall rules or not.

The *firewall implementation testing* approach evaluates the correspondence of firewall rules with respect to the actions the firewall performs, e.g., it checks whether a rule indicates to block a packet, but the firewall illegally forwards the packet, which might be caused by a firewall implementation error .

[3] Khaled Salah, Khalid Elbadawi, Member, Raouf Boutaba, "*Performance Modeling and Analysis of Network Firewalls*", Khalifa Univ. of Sci., Sharjah, United Arab Emirates, IEEE transactions on network and service management, vol. 9, no. 1, march 2012

Brief Summary:

In this paper, an analytical queuing model based on the embedded Markov chain to study and analyze the performance of rule-based firewalls when subjected to normal traffic flows as well as DoS attack flows targeting different rule positions is presented. Work has been done towards improving the firewall performance by proposing techniques to optimise and detect misconfigurations in the firewall security policies.

Two optimizing approaches have been proposed on using Ternary Content Addressable Memories(TCAM)- TCAM chip is a hardware chip dedicated for fast packet classification.

A finite queuing model with multi-stage service is used where incoming packets get queued in an Rx DMA ring. An interrupt is generated to notify the device driver of the reception of a new packet. The device driver starts executing Data Link layer functionalities and then invokes the kernel IP processing task. The kernel packet processing is responsible for performing IP Network layer. This model represents a rule- based network firewall.

To validate the model, it was subjected to two types of traffic - (1) normal traffic, (2) DDoS traffic. Consequently, the throughput and the CPU utilization have been analysed.

It was demonstrated that targeting rules at the bottom of a relatively large ruleset can be severely detrimental to the performance of the firewall. As a good design practice and vital countermeasure against DoS attacks that target bottom rules, it is recommended to minimize the size of the firewall ruleset or to rearrange dynamically rules so that bottom rules can be served at the top of the ruleset, thereby making it harder to launch such complexity algorithmic attacks that target bottom-rules.

[4] Alex X Lieu, "*Change-Impact Analysis of firewall policies*", ESORICS 2007, LNCA 4734,pp 155-170, Springer-Verlag Berlin Heidelberg 2007.

Brief Summary:

In this paper the theory and algorithms for firewall policy change-impact analysis are presented. The algorithms take as input a firewall policy and a proposed change, then output the accurate impact of the change. Thus, a firewall administrator can verify a proposed change before committing it.

The firewall change-impact analysis algorithms are implemented, and tested on both real-life and synthetic firewall policies. The experimental results show that the algorithms are effective in terms of ensuring firewall policy correctness and efficient in terms of computing the impact of policy changes.

Four types of firewall policy changes have been identified: rule deletion, rule insertion, rule modification, and rule swap. For each type of change, there is a theorem that states the decisions of which packets will be changed due to the policy change. Using the algorithms, an administrator can verify a proposed change before committing it.

Methods for correlating the impact of a firewall policy change and the high level security requirements that the firewall needs to satisfy as well as methods for making corrections if the impact of a change is not desirable are also presented in the paper.

[5] Ian Mothersole and Martin J. Reed, " *Optimising Rule Order for a Packet Filtering Firewall*", University of Essex, Wivenhoe Park, Colchester, Essex, CO4 3SQ, United Kingdom, Network and Information Systems Security (SAR-SSI), 2011 Conference 18-21 May 2011

Brief Summary:

A heuristic approximation algorithm that can optimise the order of firewall rules to minimise packet matching is presented. This paper proposes an algorithm that is designed to give good performance in terms of minimising the packet matching cost of the firewall. The performance of the algorithm is related to complexity of the firewall rule set and is compared to an alternative algorithm demonstrating that the algorithm here has improved the packet matching cost in all cases.

Firewall filtering rules may not be disjoint, which means that a packet could potentially match more than one rule. The rules could be related or dependent or in a conflict. Dependent rules have a precedence order.

The firewall optimisation problem is to place the rules in such an order so that the most frequently used rules are near to the top of the rule set and therefore reduce the packet-rule searching time. To facilitate this, the rules are associated with a weight that is equal to the number of matches of this rules in a representative flow of traffic. The optimisation problem is very similar to the popular job scheduling problem.

The proposed algorithm turned out to have an improved cost performance because it guaranteed to perform a first- order check for optimum dependent rule position. The opportunity for optimisation arises from the fact that traffic distributions and rule ranges are not uniform.

The algorithm presented here is shown to have improved performance compared to an earlier reported algorithm in all cases, at the cost of higher runtime complexity. The increased runtime complexity is unlikely to be significant for the offline optimisation of a firewall, which is the main target of this work.

[6] Dmifty Rovniagin, Avishai Wool, " *The Geometric Efficient Matching Algorithm For Firewalls*", School of Electrical Engineering, Tel Aviv University, Ramat Aviv 69978, Israel, Dependable and Secure Computing, IEEE Transactions on (Volume:8 , Issue: 1) Jan.-Feb. 2011

Brief Summary:

Firewall packet matching can be viewed as a point location problem: Each packet (point) has 5 fields (dimensions) which need to be checked against every firewall rule in order to find the first matching rule. In this paper packet matching algorithm, which we call the Geometric Efficient Matching (GEM) algorithm is presented. The GEM algorithm enjoys a logarithmic matching time performance, easily beating the linear time required by the naive matching algorithm. However, the algorithm's theoretical worst-case space complexity is **$O(n^4)$** for a rule-base with n rules.

The firewall packet matching problem finds the first rule that matches a given packet on one or more fields from its header. The packet header contains the protocol number, source and destination address and port numbers fields. First, we check the protocol field and go to the protocol array of the search data structure, to select the corresponding protocol database header. From this point, we apply a binary search with the corresponding field value on every level, in order to find the matching simple range and continue to the next level. The last level will supply us with the desired result-the matching rule number.

From the paper, it is concluded that GEM maintained a 100% throughput at all the send rates and for all rule-base sizes tried, therefore, it is an efficient rule filtering algorithm.

PROJECT BACKGROUND

Packet filtering is a network security mechanism that works by controlling what data can flow to and from a network.

To transfer information across a network, the information has to be broken up into small pieces, each of which is sent separately. Breaking the information into pieces allows many systems to share the network, each sending pieces in turn. In IP networking, those small pieces of data are called *packets*. All data transfer across IP networks happens in the form of packets.

The basic device that interconnects IP networks is called a *router*. Packets traversing an internetwork (a network of networks) travel from router to router until they reach their destination.

Packet filters act by inspecting the "packets" which transfer between computers on the Internet. If a packet matches the packet filter's set of rules, the packet filter will drop (silently discard) the packet, or reject it (discard it, and send "error responses" to the source).

This type of packet filtering pays no attention to whether a packet is part of an existing stream of traffic (i.e. it stores no information on connection "state"). Instead, it filters each packet based only on information contained in the packet itself (most commonly using a combination of the packet's source and destination address, its protocol, and, for TCP and UDP traffic, the port number).

Packet Filtering Firewalls work on the Basis of Rules defines by **Access Control Lists**. They check all the Packets and screen them against the rules defined by the Network Administrator as per the ACLs. If in case, any packet does not meet the criteria then that packet is dropped and Logs are updated about this information.

Administrators can create their ACLs on the basis Address, Protocols and Packet attributes.

Functions of a Packet Filtering Firewall:

- **Control:** Allow only those packets that you are interested in to pass through.
- **Security:** Reject packets from malicious outsiders
- **Watchfulness:** Log packets to/from outside world

Important Features of Packet Filters

- The firewall should provide good deal of *logs*. The more detailed are the logs, the better the protection.
- The command line syntax or GUI of firewall should be *easy to create new rules* and of course firewall exceptions.
- The packet filter *orders* should be evaluated carefully in order to make the filtering fruitful.

Different types of filtering mechanisms:

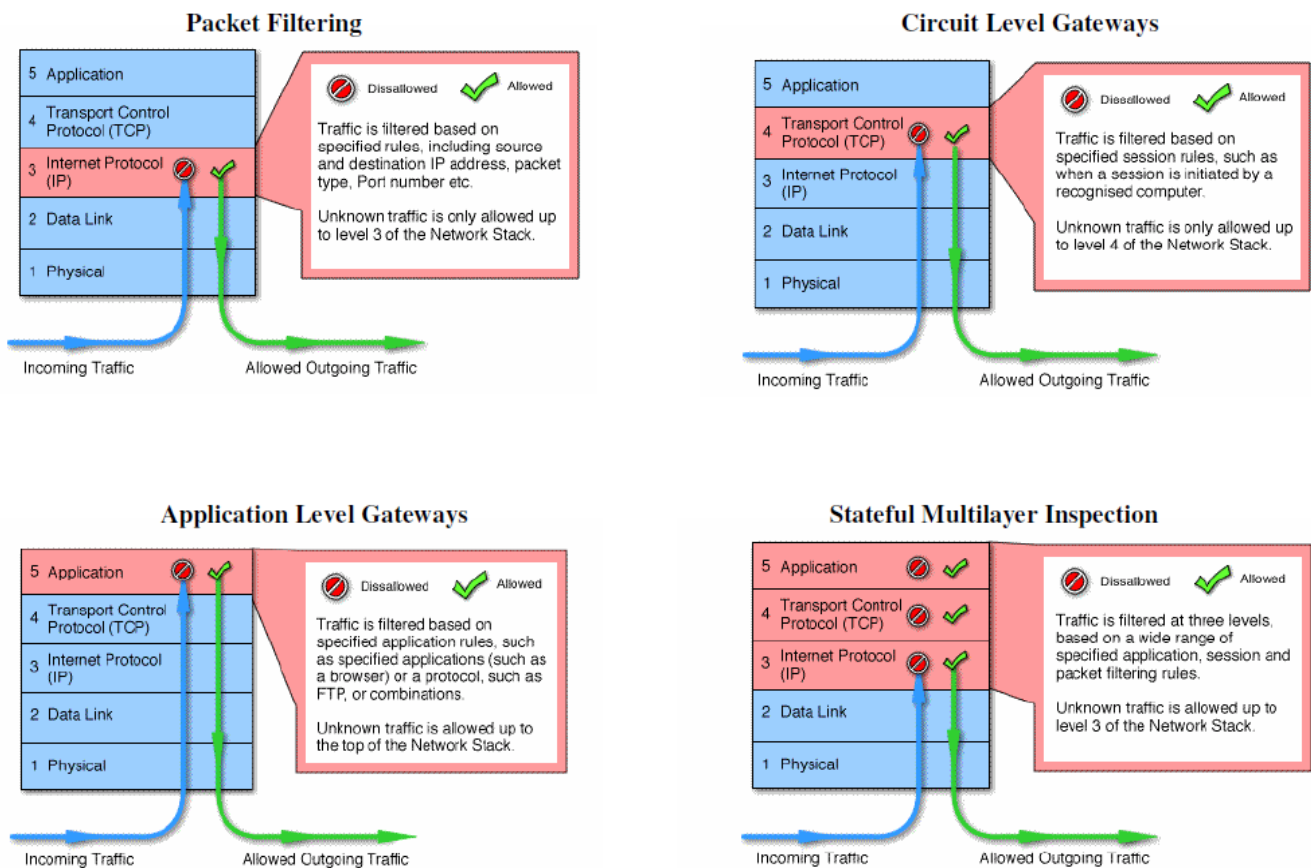


Fig. 4

Packet filters –

- work at the network level.
- compared to a set of criteria before it is forwarded

Advantages: low cost, low impact on network performance.

Disadvantages: does not support sophisticated rule based models.

Circuit level gateways –

- work at the session layer
- monitor TCP handshaking between packets to determine whether a requested session is legitimate
- Information passed to remote computer through a circuit level gateway appears to have originated from the gateway.

Advantages: relatively inexpensive , hiding information about the private network

Disadvantages: they do not filter individual packets.

Application level gateways –

- work at the application layer
- Incoming or outgoing packets cannot access services for which there is no proxy
- filter application specific commands
- can also be used to log user activity and logins.

Advantages: a high level of security

Disadvantages: having a significant impact on network performance, not transparent to end users and require manual configuration of each client computer.

Stateful multilayer inspection firewalls–

- work at the application , session, network layer.
- They filter packets at the network layer, determine whether session packets are legitimate and evaluate contents of packets at the application layer
- They allow direct connection between client and host, alleviating the problem caused by the lack of transparency of application level gateways. can also be used to log user activity and logins.
- They rely on algorithms to recognize and process application layer data instead of running application specific proxies.

Advantages: a high level of security, good performance, transparency to end users

Disadvantages: they are expensive and complex.

PROJECT DESIGN

Model used

Incremental Model

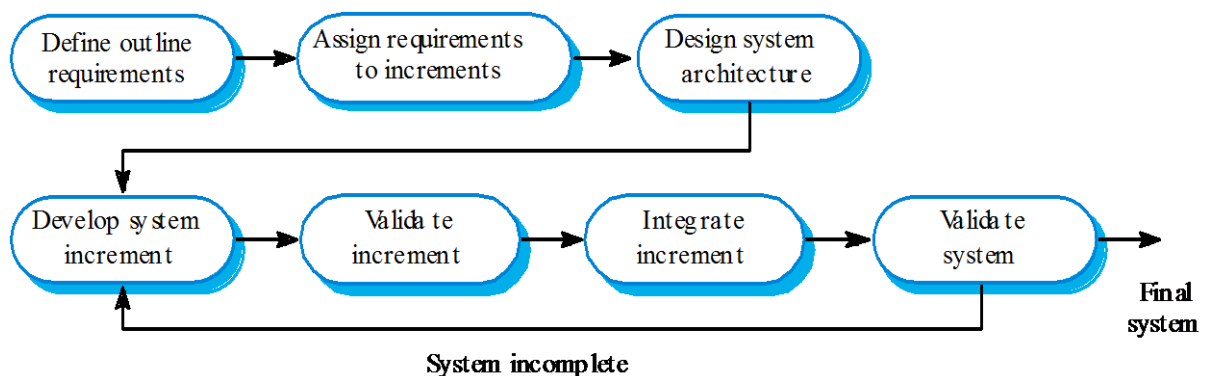


Fig. 1

The Incremental Model is an evolution of the waterfall model, where the waterfall model is incrementally applied.

The series of releases is referred to as “increments”, with each increment providing more functionality to the customers. After the first increment, a core product is delivered, which can already be used by the customer. Based on customer feedback, a plan is developed for the next increments, and modifications are made accordingly. This process continues, with increments being delivered until the complete product is delivered.

Advantages

- It is generally easier to test and debug than other methods of software development because relatively smaller changes are made during each iteration. This allows for more targeted and rigorous testing of each element within the overall product.
- Customer can respond to features and review the product for any needful changes.
- Initial product delivery is faster and costs lower.

Disadvantages

- Resulting cost may exceed the cost of the organization.
- As additional functionality is added to the product, problems may arise related to system architecture which were not evident in earlier prototypes.

Input/ Output:

- **Input:**

Packets coming from network

- **Output:**

List of blocked packets and their IP addresses and the packets that passed the firewall successfully.

Flow Chart:

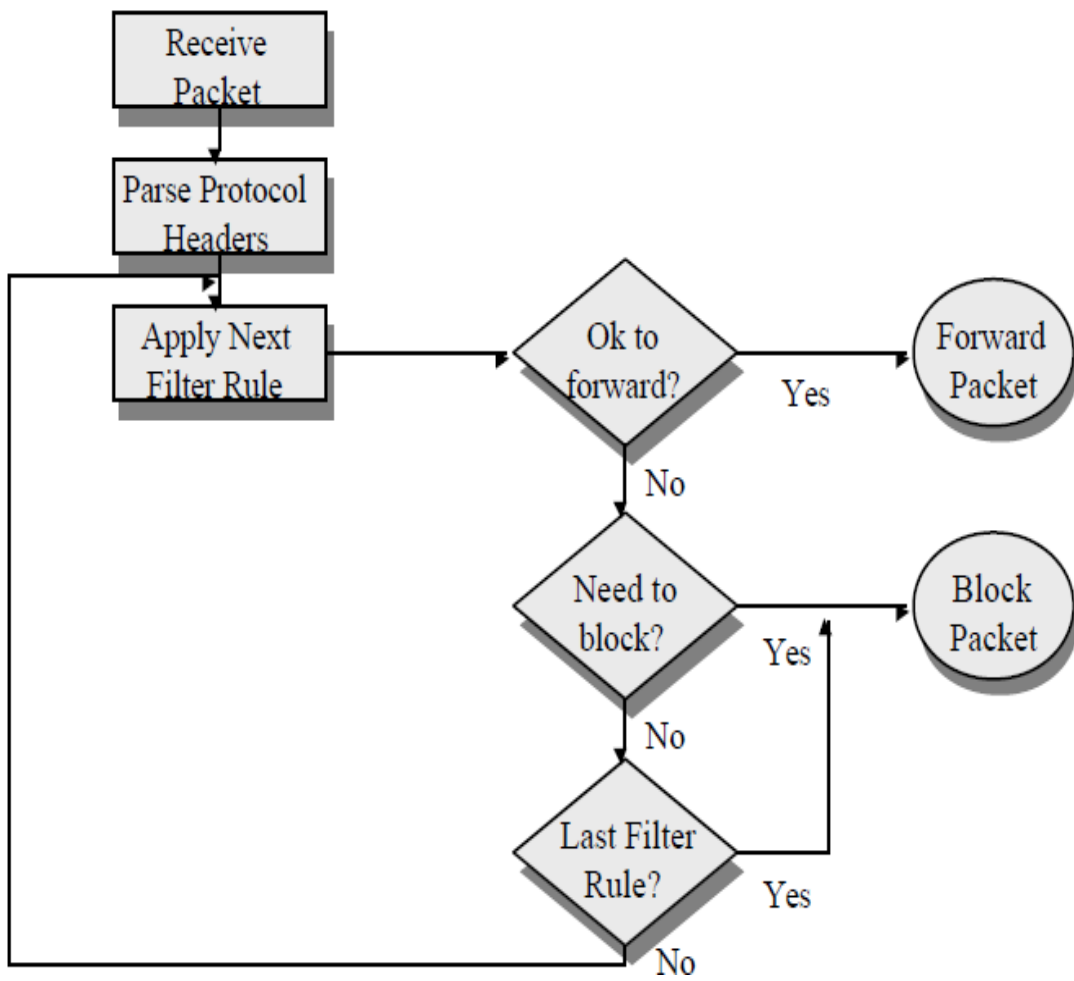


Fig. 2

Use Case diagram:

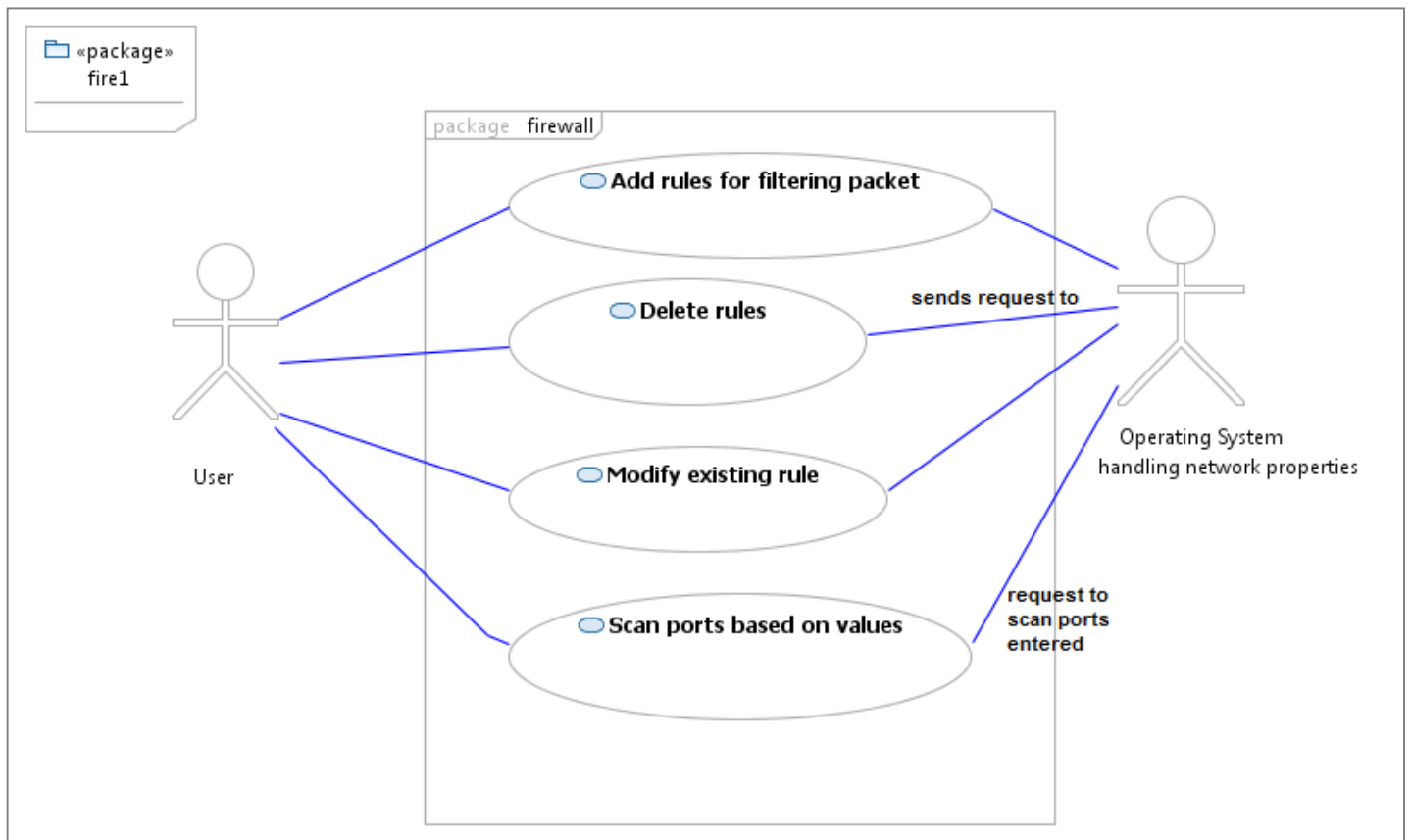


Fig. 3

Working of the filter:

Working of firewall is based on the following steps:

- Extract the packet header
- Check the protocol associated
- Compare with the rules
- Check the source and destination add. If protocol is same
- Check out the port if protocol is TCP
- Drop or pass the packet

Brief description:

After declaring structure variables, an integer type “countrule” is declared and initialized it holds the value of the number the rule, it is incremented when new rule is required. Filterlist is initialized to first, its size increases as more and more rules are added. Now the packet header is extracted and is assigned to the variable `ipp`.

Next the protocol is checked.

If the protocol is numbered as 6 means it is TCP. We accept all the packets if the connection is already established. Also if we don't have the bit SYN activate then we pass the packet by using `return PF_FORWARD`. Otherwise the packet is compared against the rules from the list until there is no member is in the list means till the condition `while (aux! =NULL)` persists. Now check if the protocol is same, if it is then look for the source and destination address and each time increment the countrule. Now if the protocol is TCP check for the port.

Now the decision can be taken whether to drop or pass the packet according to the following statements

If (`aux->ipf.drop`)

`return PF_DROP; //drop the packet`

else

`return PF_FORWARD; //forward the packet`

The same procedure is done for the packets of the UDP protocols.

Description of the classes used:

Module 1

Administrator class:

This is the first driving class of the program that will accept the user name and the password. Only if the user supplies the correct entries the software will enable the user to proceed ahead otherwise not

- Return value:

On success: MainMenu() will be called.

On failure: exit() will execute.

- Parameter:

User ID and Password.

- Pseudocode:

BEGIN:

1. [DECLARE] command CHARACTER 20;
2. command=GETCOMMAND();
3. IF(command= ("OK")) THEN
{
 IF(username= (userid.GETTEXT()) AND pwd= (password.GETTEXT()))
 THEN
 MainMenu();
 }
4. ELSE
PRINT("Invalid Username OR Password ! Reenter the Username & Password ");
 END:

Module 2

ftpFilter class

ftpFilter class is used to perform various actions on a ftp file. This class first performs a check to an ftp file, If the file passes the firewall barrier then the file is stored. This class also provides methods to create dialog boxes which prompt the user to provide permission to perform various actions on the ftp file. The various actions include reading, writing, renaming, deleting, traversing and listing the ftp file. When the user selects the desirous option by marking on the checkbox and clicking on OK an event is fired and corresponding action is performed on the ftp file.

Module 3

httpFilter class

By using this class the user defines if he allows flash files, java applets, java scripts or VB scripts to be or not to be accessed by the user on a http file.

Module 4

sessionViewer class

The session Viewer class shows in a tabular manner the internal and external source of the ongoing connection or the history of connections

Module 5

siteAuthen class

In this class a dialog box is created using which, the user enters either the ip address or domain name and clicks on ok. As ok is pressed an event is fired and it is handled. A table is also created using JTable and the information entered by the user is saved in a file and shown in the table.

Module 6

siteLocker class

This class provides the site locker feature to our firewall. This class is used to create a dialog box where the user specifies the site (to be blocked)in the text box by either entering the domain name or the ip address, As the user enters the information and hits on ok an event is fired which is handled. The site or ip address is then read and stored in a file. For the convenience of the user this class also provides a table in that dialogue box wherein the previously blocked site or ip addresses also appear.

Module 7

keywordlocker class

Using this filter the administrator of the firewall can block particular keywords that he doesn't want to be seen by the users. The domain name that has this keyword as its part cannot be accessed. It can also be the domain name that represents multiple computers.

Module 8

ftpproxy class

To allow various actions through FTP Server.

Module 9

mainMenu class

This class draws the main screen of the software. It allows the administrator to configure the various settings. Depending on the kind of setting the program flow will move on.

- Return value:
This class may pass the control to either Server class, FtpProxy class, SessionViewer class, About class, Help class or exit.
- Pseudo code:
BEGIN:
 1. [DECLARE] j CHARACTER 20
 2. IF(j= =About)
 CALL About();
 3. IF(j= =Help)
 CALL Help();
 4. IF(j= =http)
 CALL http();
 5. IF(j= =ftp)
 CALL ftp();
 6. IF(j= =SessionViewer)
 CALL SessionViewer();
 7. IF(j= =exit)
 CALL exit();END:

Module 10

serialization class

It is used for saving the state of an Object, and being able to reconstitute that state at a later time

Module 11

about class

it displays the firewall version, authors and contact email id.

Module 12

help class

It is designed to make the firewall more user-friendly as it represents all the topics in a tree form.

Module 13

server class

This class checks the protocol of incoming site converts it and save the packet of data into a file (as per the rule base of our firewall the site is allowed to be accessed.). This class saves the number of sockets, ports and connections. It used the concept of threads.

An object of Session Viewer class is made and it is continuously passed data from the session viewer class. An object of statistic class (serialization class is made) and this object is used to read the file. Protocol check is provided here.

An array of server socket, client socket and no of thread is made and no of ports defined the size of these arrays. In the array of server socket port number and number of connections are given as input to it. The number of threads is equal to the number of ports

Server Socket

This class implements server sockets. A server socket waits for requests to come in over the network. It performs some operation based on that request, and then possibly returns a result to the requester.

Client socket

```
cs[index]=ss[index].accept()
```

using the above command the requested file is delivered on the client socket

```
accept()
```

this function accepts a connection on a socket. An incoming connection is acknowledged and associated with an immediately created socket.

Now this client socket's index is buffered and stored into a buffer named inp, now this inp is in turn read an stored into a String type object called line. Now this line string stores the url.

To get the path of this url we use this function

```
value=line.substring(line.indexOf(":")+2,line.length())
```

Another string key is created which is implemented as

```
key=line.substring(0,line.indexOf(":"))
```

Now in a hash table the key value is stored and the above details is stored and written in a file and output to the user in a tabulated form.

Now a check is provided on every client port. The path is checked with the validation of protocol , keyword check and a list of allowed sites specified by the user. If this check returns true then the user is allowed to access this file else the site is blocked. For allowing a connection .openConnection() method is used.

Module 14

Close Dialog Box

This class prompts the user to exit from the firewall server. When the user clicks on exit a window appears which ask to exit from the firewall serer if the user clicks 'ok' then it exits or if the user clicks cancel it will not exit.

[D] Various Pre Defines Methods, Classes, Events, Handlers and Interfaces used:

- JDialog Interface

Imported from swing package, it is a class for creating a dialog window. You can use this class to create a custom dialog, or invoke the many class methods to create a variety of standard dialogs.

-. ActionListner event

The listener interface for receiving action events. The class that is interested in processing an action event implements this interface, and the object created with that class is registered with a component, using the component's addActionListener method. When the action event occurs, that object's **actionPerformed** method is invoked.

A semantic event which indicates that a component-defined action occurred. This high-level event is generated by a component (such as a Button) when the component-specific action occurs (such as being pressed). The event is passed to every every ActionListener object that registered to receive such events using the component's **addActionListener** method.

Syntax: void **actionPerformed(ActionEvent e)**

Invoked when an action occurs.

-.KeyListener event

The listener interface for receiving keyboard events (keystrokes)

Syntax: void **keyPressed(KeyEvent e)**

Invoked when a key has been pressed.

- **JPanel**

It is a generic light weight container with a double buffer and a flow layout.

- **ProgressMonitor**

A class to monitor the progress of some operation. If it looks like the operation will take a while, a progress dialog will be popped up. When the ProgressMonitor is created it is given a numeric range and a descriptive string. As the operation progresses, call the setProgress method to indicate how far along the [min,max] range the operation is. Initially, there is no ProgressDialog. After the first millisToDecideToPopup milliseconds (default 500) the progress monitor will predict how long the operation will take. If it is longer than millisToPopup (default 2000, 2 seconds) a ProgressDialog will be popped up.

- **.setForeground()**

for setting the foreground

- password.**setEchoChar(*)**;

JPasswordField is intended to be source-compatible with java.awt.TextField used with echoChar set. It is provided separately to make it easier to safely change the UI for the JTextField without affecting password entries.

- **actionPerformed**

This method of JDialog interface is over ridden(method is used to check the password and usrid)

- **System.exit()**

Terminates the currently running Java Virtual Machine

- **textfield**

A TextField object is a text component that allows for the editing of a single line of text.

- **font**

The Font class represents fonts, which are used to render text in a visible way.

-JLabel

A display area for a short text string or an image, or both.

-setMnemonic()

to associate mnemonics to menu items. Mnemonics allows user to interact with menu items using keys on keyboard.

-Setfocuspainted()

Sets whether focus should be painted.

-Parameters():

If true, the focus state is painted.

-setBorderPainted()

method is used to decide whether the progress bar should paint its border or not. The default value is false: progress bar without border.

-setRolloverIcon

used to set the icon or image to a button for display when the mouse pointer rolls over the icon or the button.

-setPressedIcon(ImageIcon)

method to set a pressed icon in the button.

-setactioncommand()

Returns the command name of the action event fired by this button. If the command name is null (default) then this method returns the label of the button.

-setToolTipText

method to set up a tool tip for the component.

-actionPerformed(ActionEvent e)

Invoked when an action occurs.

-keyPressed(KeyEvent e)

Invoked when a key has been pressed.

-keyReleased(KeyEvent e)

Invoked when a key has been released.

-keyTyped(KeyEvent e)

Invoked when a key has been typed.

-ImageIcon

An implementation of the Icon interface that paints Icons from Images.

-JTextField

is a lightweight component that allows the editing of a single line of text.

-JButton

An implementation of a "push" button.

- JTable

is used to display and edit regular two-dimensional tables of cells.

-A generic Abstract Window Toolkit(AWT) container object is a component that can contain other AWT components.Components added to a container are tracked in a list.

-setForeground(Color fg)

Sets the foreground color of this component.

-setActionCommand(String command)

Sets the command name for the action event fired by this button.

-setLocation(int x, int y)

Changes the point to have the specified location.

-JFrame

adds support for the JFC/Swing component architecture.

-Jmenu

An implementation of a menu -- a popup window containing JMenuItem objects that is displayed when the user selects an item on the JMenuBar.

-Jmenubar

An implementation of a menu bar. You add JMenuItem objects to the menu bar to construct a menu. When the user selects a JMenuItem object, its associated JPopupMenu is displayed, allowing the user to select one of the JMenuItem objects on it.

-setSize(Dimension d)

Resizes this component so that it has width d.width and height d.height.

-setVisible(boolean b)

Shows or hides this component depending on the value of parameter b.

-setTitle

Defines the title of the document.

-setResizable(boolean resizable)

If the parameter is false then the user cannot re-size the frame.

-Runnable

The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread. The class must define a method of no arguments called run.

-ServerSocket

This class implements server sockets. A server socket waits for requests to come in over the network. It performs some operation based on that request, and then possibly returns a result to the requester.

-Socket

This class implements client sockets (also called just "sockets"). A socket is an endpoint for communication between two machines.

-BufferedReader

Reads text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines.

-Writer

Abstract class for writing to character streams. The only methods that a subclass must implement are write(char[], int, int), flush(), and close().

-getInetAddress()

Returns the address to which the socket is connected.

\-gethostaddress()

Returns the Internet Protocol (IP) addresses for the specified host.

-gethostname function

retrieves the standard host name for the local computer.

-InputStreamReader

An InputStreamReader is a bridge from byte streams to character streams: It reads bytes and decodes them into characters using a specified charset.

-OutputStreamWriter

An OutputStreamWriter is a bridge from character streams to byte streams: Characters written to it are encoded into bytes using a specified charset.

-getOutputStream()

Gets the output stream of the subprocess.

-readLine()

Read a line of text.

-flush()

Flushes the output stream and forces any buffered output bytes to be written out.

-setValueAt

Sets the value in the cell at columnIndex and rowIndex to aValue.

-Calendar

provides methods for converting between a specific instant in time and a set of calendar fields such as YEAR, MONTH, DAY_OF_MONTH, HOUR.

\-getInstance()

Gets a calendar using the default time zone and locale.

-ObjectInputStream

De-serializes primitive data and objects previously written using an ObjectOutputStream.

-getContentPane()

It is a method that returns a container that is where you usually add all of your components instead of adding Component objects directly to a window such as a JFrame or a JDialog.

-setLayout

Sets or gets the layout manager for this panel. The layout manager is responsible for positioning the panel's components within the panel's bounds according to some philosophy.

-setBorder

To put a border around a JComponent, we use setBorder method.

-setBackground

to set the color and the background.

-treeexpnasion listener

Called whenever an item in the tree has been expanded.

-setBounds(int x,int y,int width,int height)

positions the called object in the specified location and in specified size.

-window listener and window adapter

The listener interface for receiving window events. The class that is interested in processing a window event either implements this interface (and all the methods it contains) or extends the abstract WindowAdapter class (overriding only the methods of interest). The listener object created from that class is then registered with a Window using the window's addWindowListener method. When the window's status changes by virtue - the event responsible for the update of being opened, closed, activated or deactivated, iconified or deiconified, the relevant method in the listener object is invoked, and the WindowEvent is passed to it.

-HyperlinkEvent

Is used to notify interested parties that something has happened with respect to a hypertext link.

e- the event responsible for the update

-TreeSelectionMode

This interface represents the current state of the selection for the tree component.

-SINGLE_TREE_SELECTION

Selection can only contain one path at a time.

-TreeSelectionListener

The listener that's notified when the selection in a TreeSelectionMode changes

-TreeSelectionEvent e

Called whenever the value of the selection changes the event that characterizes the change.

-JTree()

Returns a JTree with a sample model.

-JScrollPane()

Creates an empty (no viewport view) JScrollPane where both horizontal and vertical scrollbars appear when needed.

-JEditorPane()

Creates a new JEditorPane.

-DefaultMutableTreeNode()

Creates a tree node that has no parent and no children, but which allows children.

-getContentPane()

It is a method that returns a container that is where you usually add all of your components instead of adding Component objects directly to a window such as a JFrame or a JDialog.

-KeyEvent.VK_ESCAPE

For key typed events, the getKeyCode method always returns VK_UNDEFINED (Virtual key methods)

-ItemListener events

ItemListener interface is used for receiving item events

- addActionListener()

Register an instance of the event handler class as a listener on one or more components

- addItemListener()

Adds a listener to receive item events, when the state of an item is changed by the user.

Results and Discussions.

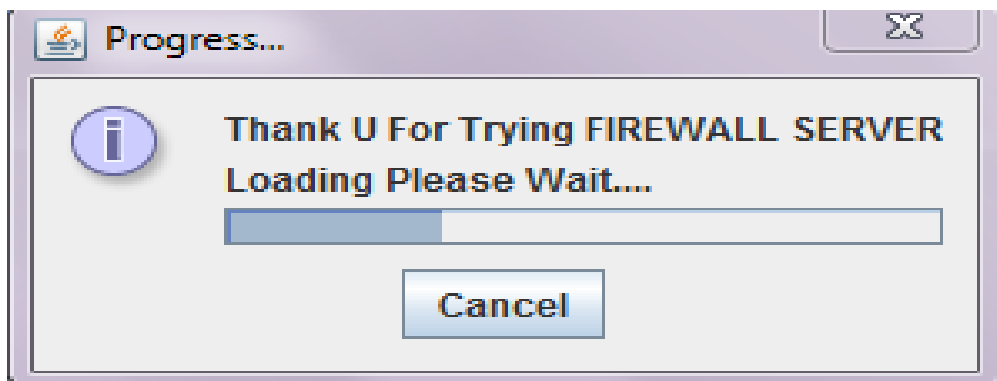
[A] Various Screenshots of the associated demo of our firewall are as follows:

The source code named firewall is compiled in cmd by following these steps:

1. Open cmd
2. Set the path for the source code
3. Compile the code by typing “javac firewall.java” in cmd
4. Now run it by typing “java firewall”.

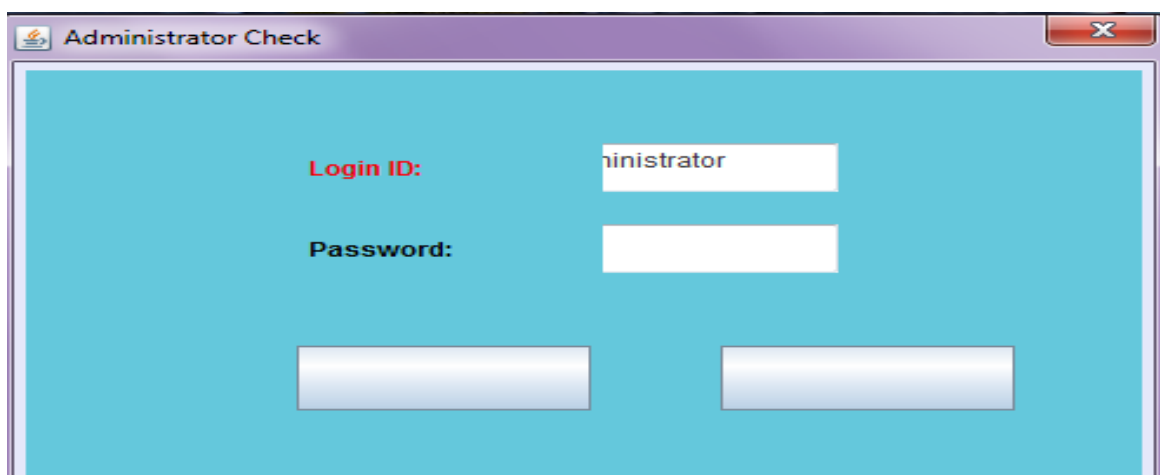
After following the above steps the following output is obtained:

- A. This is the first dialog box that appears



- B. This is the second dialog box

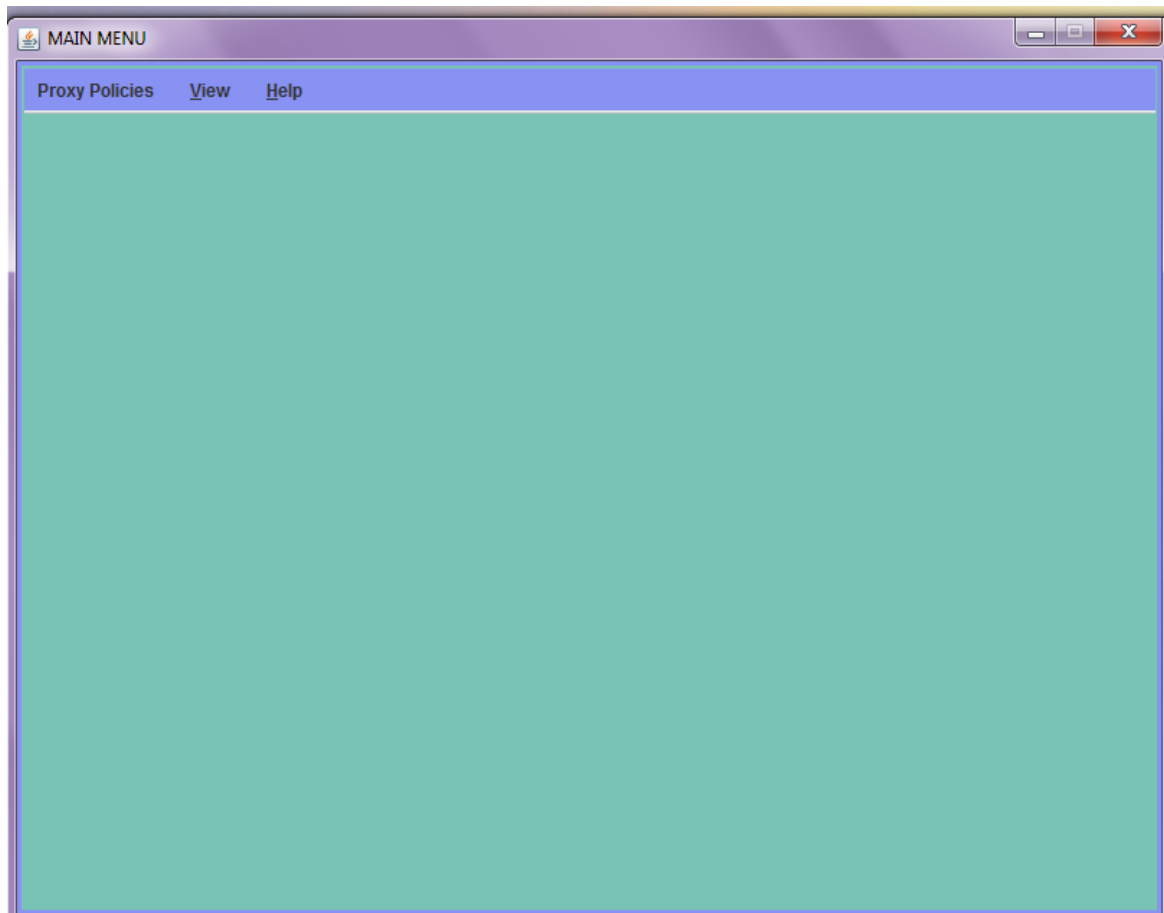
Type administrator as the user name and “anukriti” or “mehak” or “mango” as password click ok.



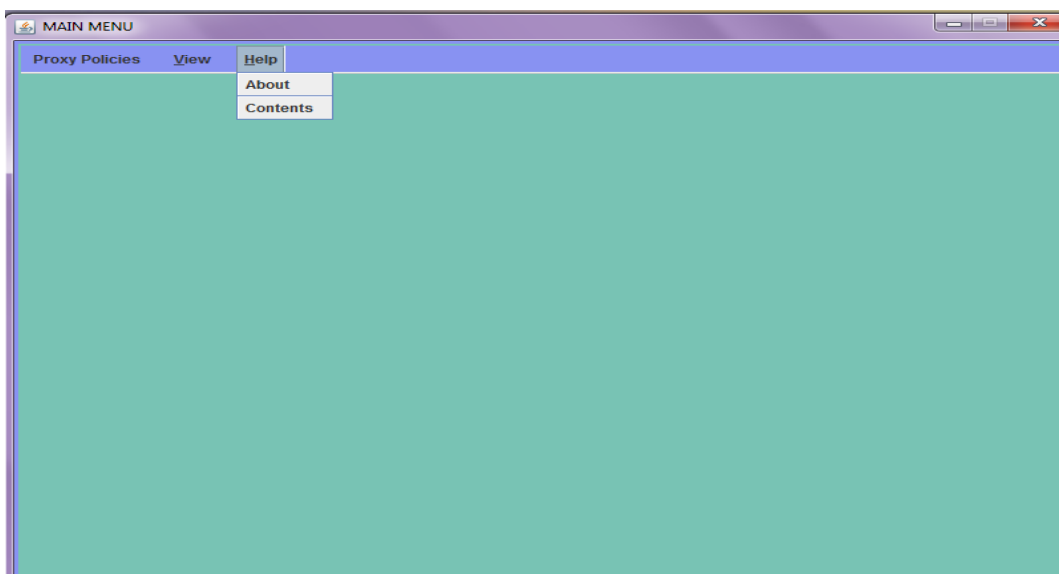
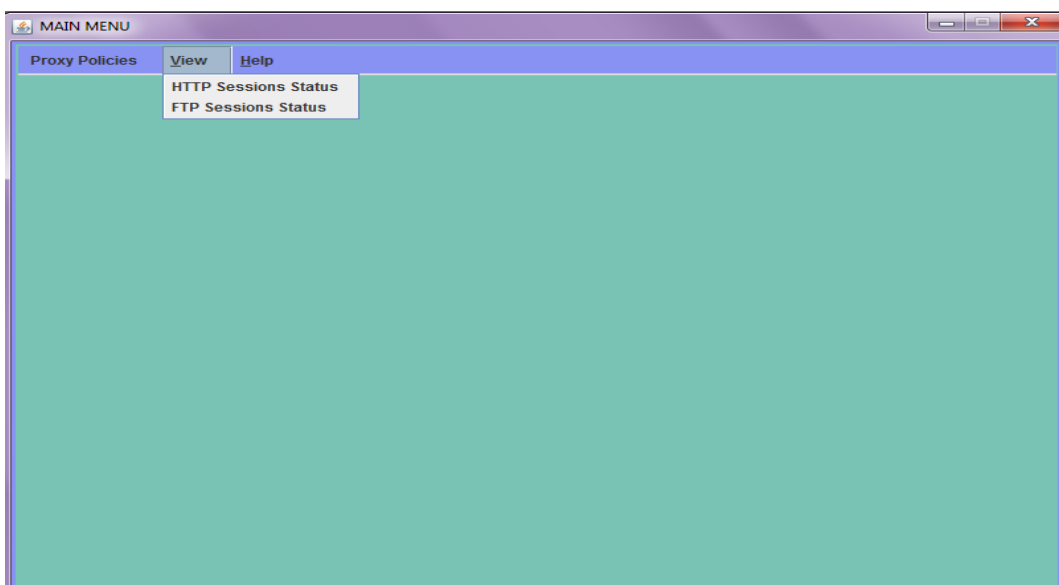
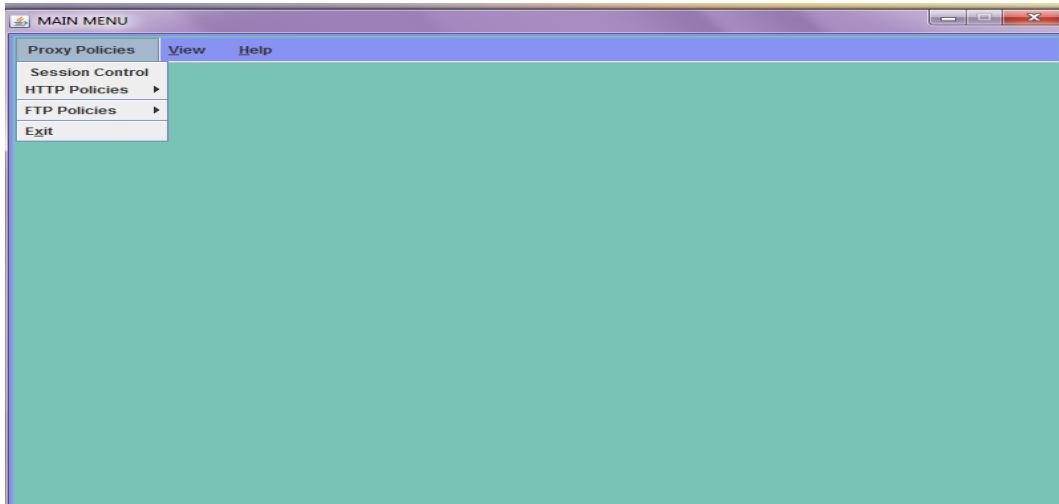
C. This is the third dialog box that pops up after log in and it shows the **main menu**.

After successful login the above screen appears.

- To start the HTTP proxy server, from Main Menu go to proxy policies-> HTTP policies-> start and view, the session viewer window will appear.
- To start the FTP proxy server, from the Main Menu go to the proxy policies-> FTP policies-> start FTP server. A dialog box asking FTP server name will appear. Enter the FTP server name and click OK, the FTP session viewer window will appear.
- After starting the proxies to view the sessions go to View-> HTTP session Status/FTP session Status. The respective session viewer window will appear.

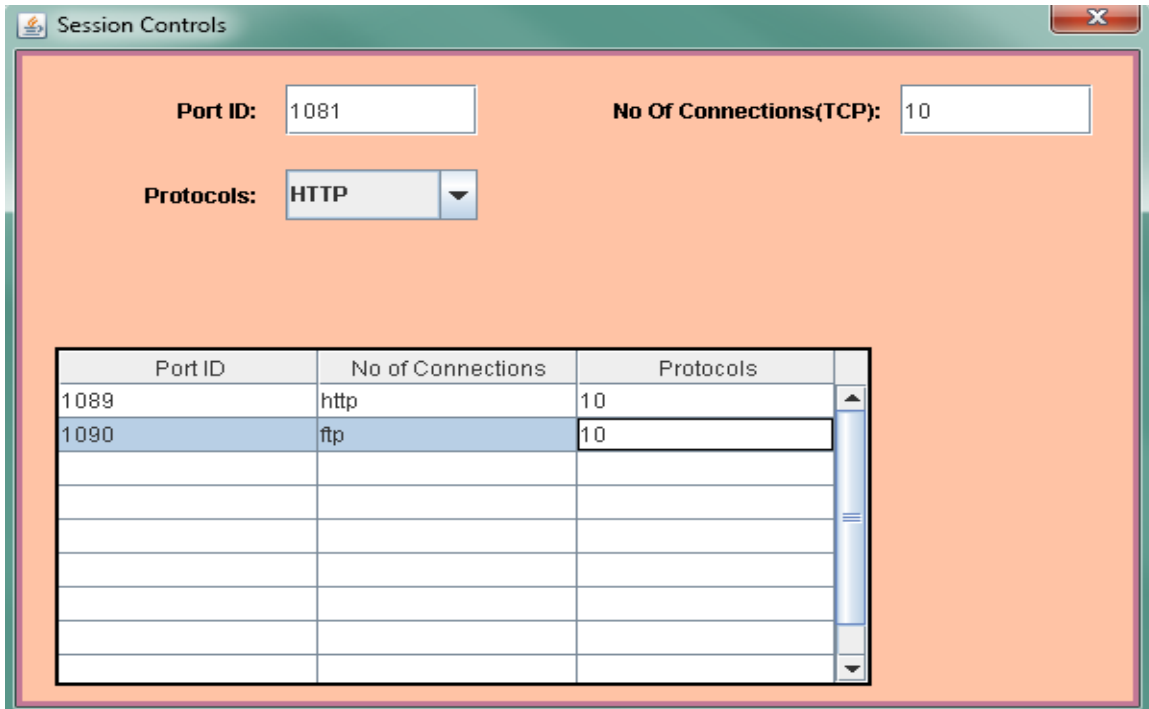


C.(a). The following 3 outputs shows the menu that get dropped down after clicking on “Proxy policies”, “View” and “Help”.

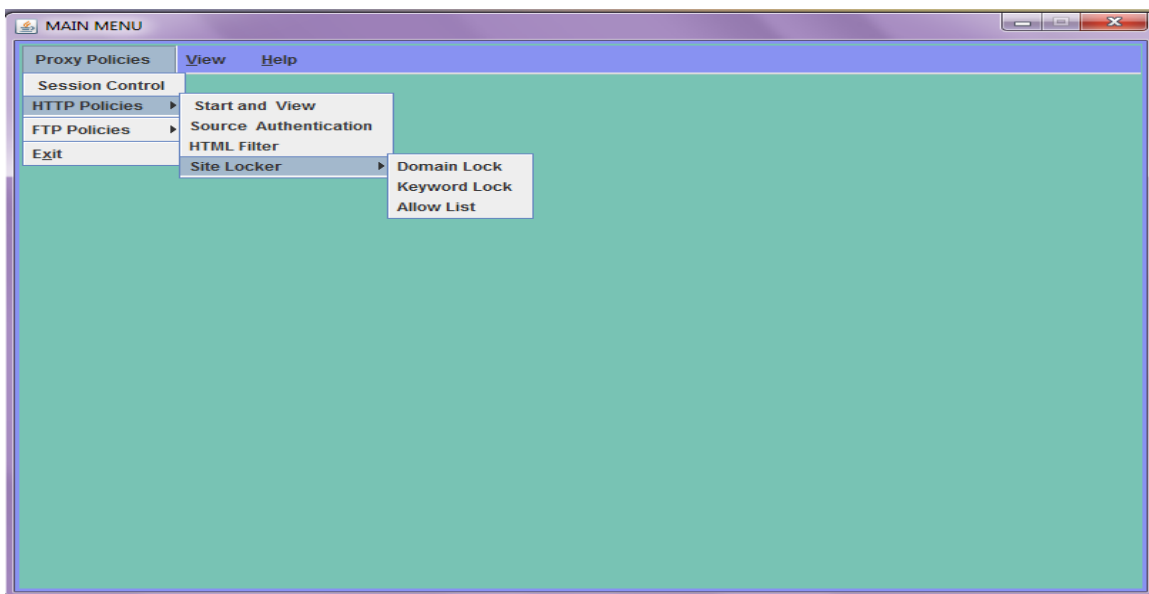


D. It shows the “**session controls**” dialog box where we can mention various details for the rule base of our firewall’s session.

- First go to Proxy policies on the menu bar then click session control.
- For creating sessions enter port ID, no. of connections, and select the protocol.
- Click add, the respective entry will appear in the table.
- Similarly for removing an entry select that entry from table and click remove.

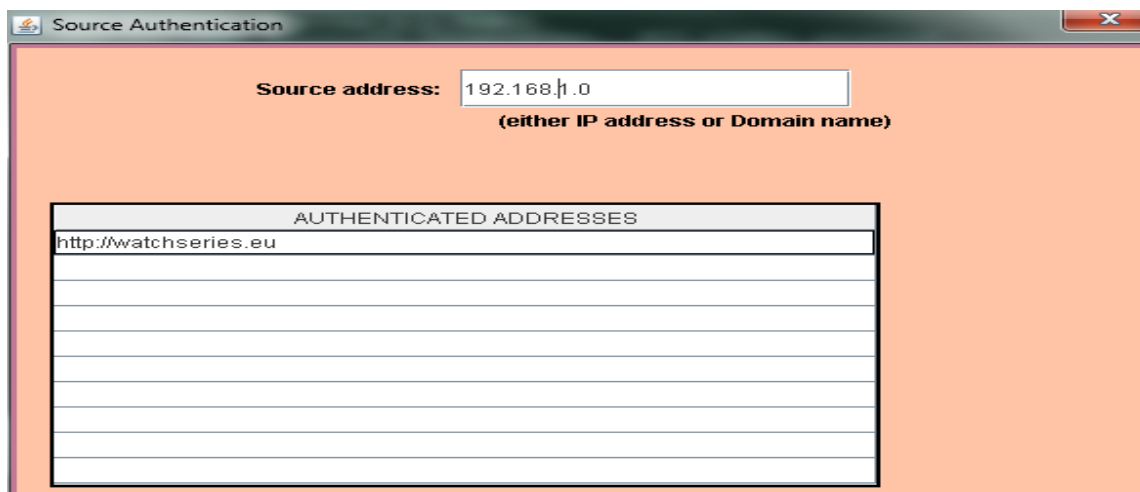


E. It shows the various sub divisions of “**proxy policies**” tab



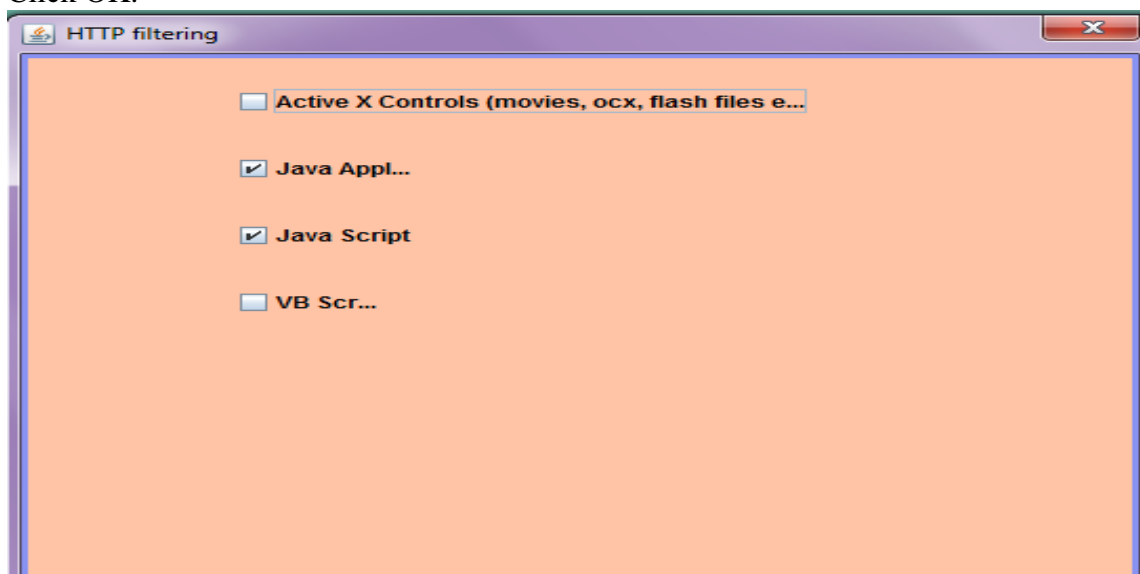
F. It shows the **source authentication** dialog box.

- First go to Proxy policies on the menu bar then click HTTP policies and click source authentication.
- For adding a new source give the source IP address in the respective text field.
- Click add, the respective entry will appear in the table.
- Similarly for removing an entry select that entry from table and click remove.



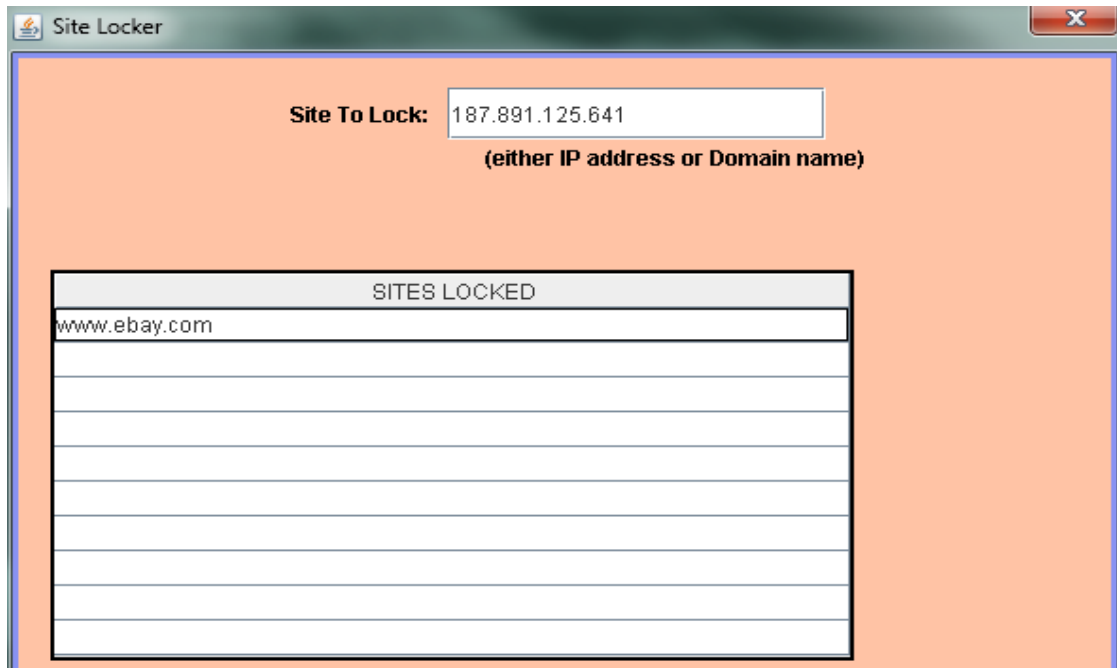
G. This is the **“http filtering”** dialog box

- From the Main Menu go to the proxy policies-> HTTP policies-> HTML filter, the above screen will appear.
- Check or uncheck the check boxes as per the filtering requirement.
- Click OK.



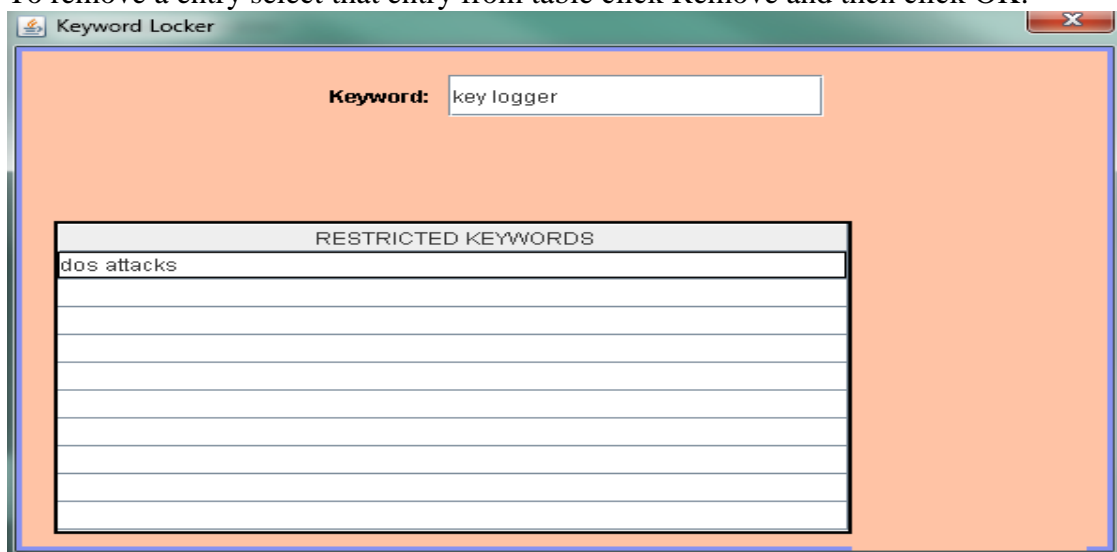
H. This is the “site locker” dialog box

- From the Main Menu go to the proxy policies-> HTTP policies-> HTML filter, the above screen will appear.
- Enter the site to lock.
- Click add to enter the site.
- Click remove to delete the entry.
- Click OK.



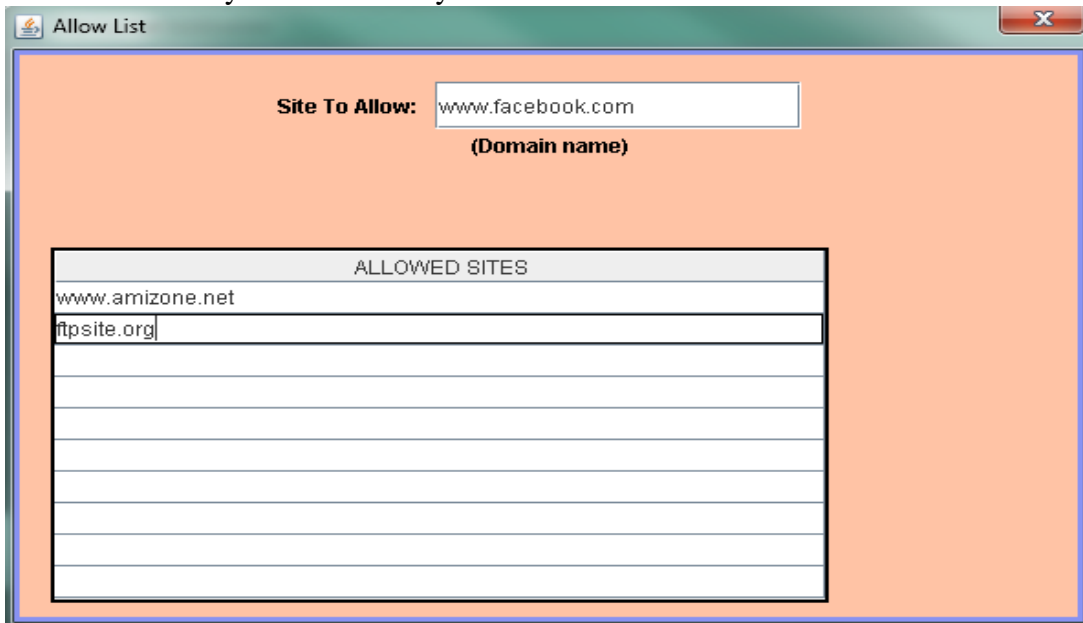
I. This is the “keyword locker” dialog box

- From the Main Menu go to the proxy policies-> HTTP policies-> site locker->keyword lock, the above screen will appear.
- Enter the keyword that has to be locked.
- Click OK.
- To remove a entry select that entry from table click Remove and then click OK.

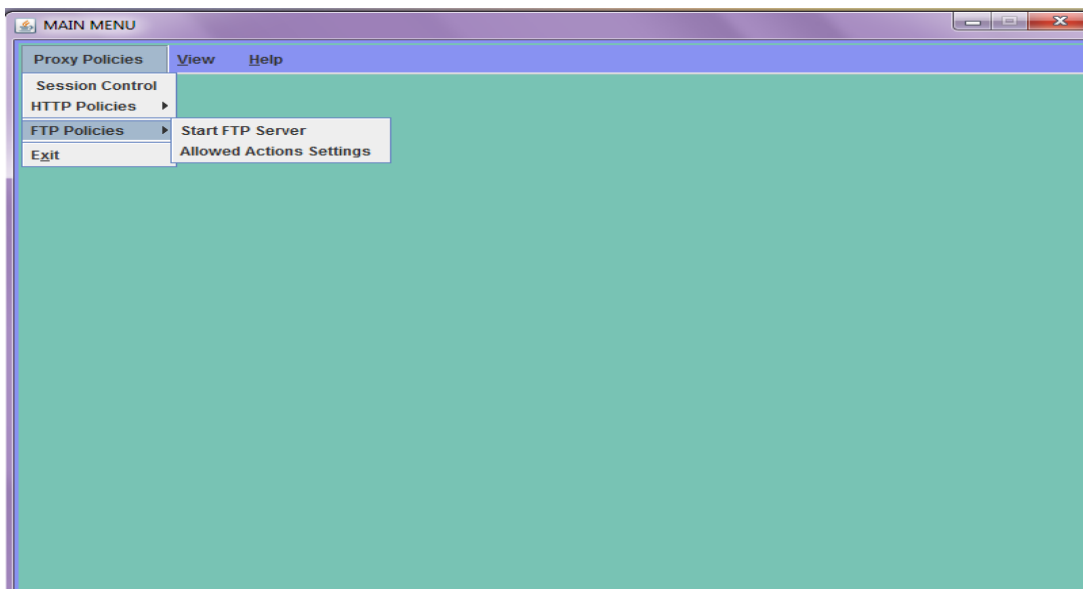


J. This is the “allow list” dialog box.

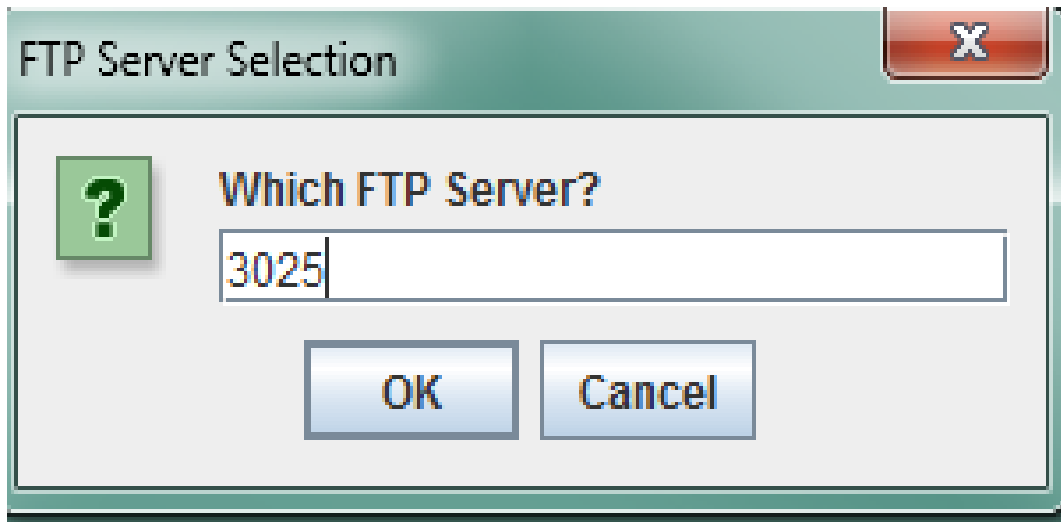
- From the Main Menu go to the proxy policies-> HTTP policies-> site locker->Allow List, the above screen will appear.
- Enter the domain name or IP address of the site that is allowed
- Click add to make an entry for the site or the domain.
- Click OK.
- To remove a entry select that entry from table click Remove and then click OK



K. This output shows the menu one gets after clicking on “ftp policies”.

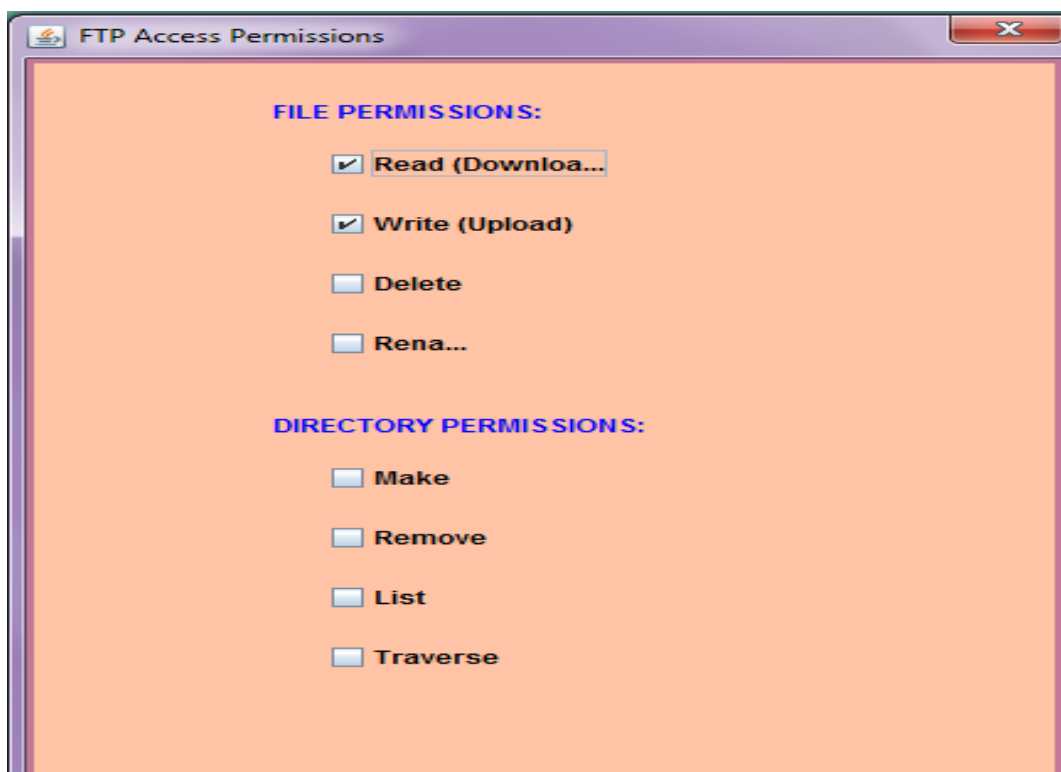


- L. This dialog box shows the **ftp server selection**. After typing the server and after clicking on this the ftp server starts.

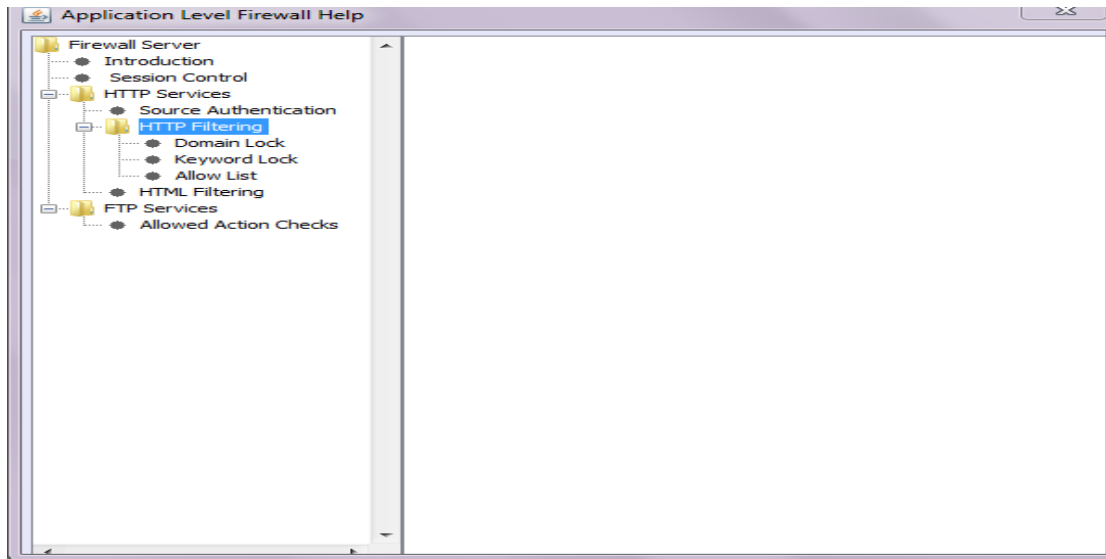


- M. It shows the **ftp Access permissions** options where the user can define all the accesses he wants to define for “ftp”.

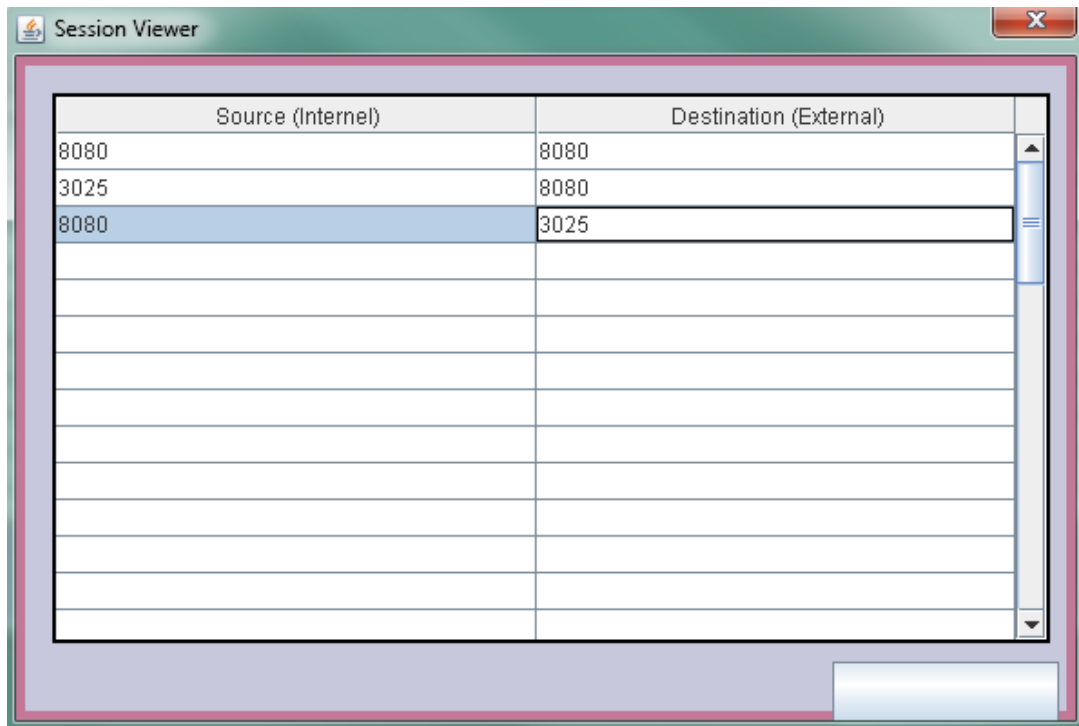
- From the Main Menu go to proxy policies-> FTP policies-> Allowed action settings, the above screen will appear.
- Check or Uncheck File permissions or directory permissions that are desired.
- Click OK.



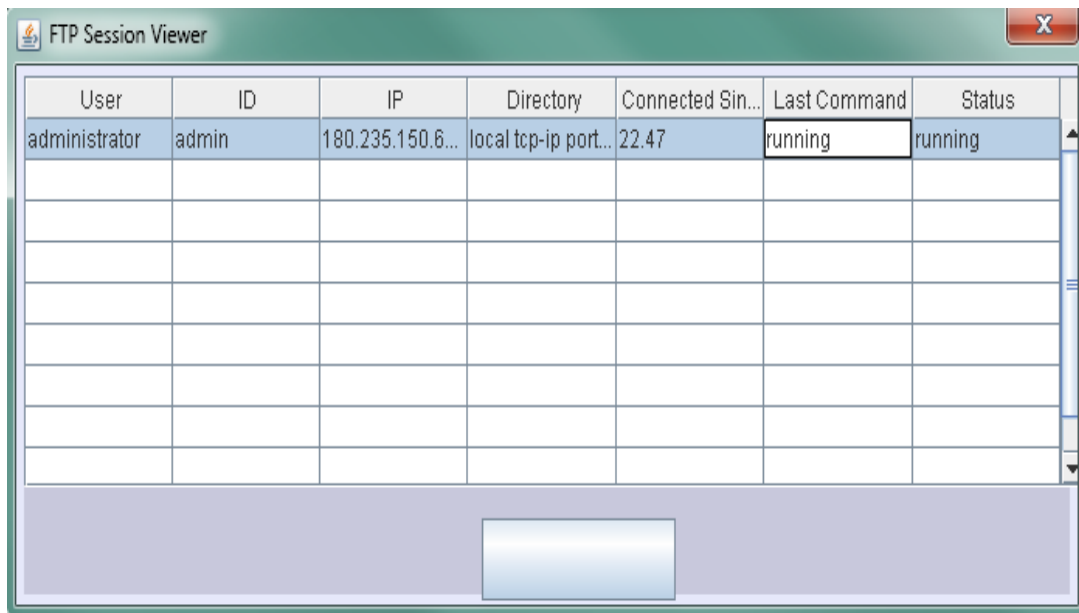
N. It shows the “help” dialog box.



O. It shows the **http session viewer** in the firewall



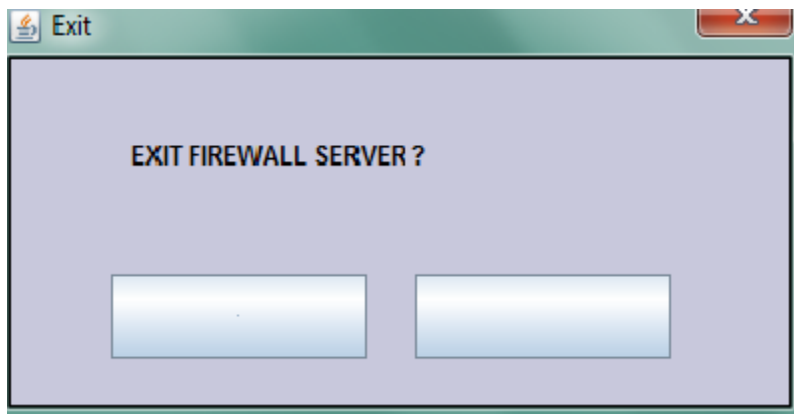
P. It shows the **ftp session viewer** in firewall



The screenshot shows a window titled "FTP Session Viewer" with a table of active sessions. The table has seven columns: User, ID, IP, Directory, Connected Sin..., Last Command, and Status. The first row contains the following data: User: administrator, ID: admin, IP: 180.235.150.6..., Directory: local tcp-ip port..., Connected Sin...: 22.47, Last Command: running, and Status: running. Below the table is a single button.

User	ID	IP	Directory	Connected Sin...	Last Command	Status
administrator	admin	180.235.150.6...	local tcp-ip port...	22.47	running	running

Q. It shows the “**close dialog box**”.



Advantage of the project to the modern age

It is a Concurrent Server so multiple requests can be handled simultaneously

- It waits for requests and grants connection if it is valid, if unexecuted threads still remain then it serves them too & after that the proxy server receives data from server & delivers it to client.
- Proper error pages are sent in case of improper request
- It makes use of JAVA which is a platform independent, robust and secure language.

CONCLUSION AND RECOMMENDATIONS

- ❖ It can be concluded that the firewall made is sufficient enough for the following:
 - ✓ Content Filtering.
 - ✓ Blocking of client IP addresses
 - ✓ Blocking of web sites.
- ❖ It is recommended that the firewall can be made more effective by distributed system architecture.

IMPLICATIONS FOR FUTURE RESEARCH

- Firewall can be extended to filter images (Use of AI)
- Firewall can be extended by implementation of cryptographic protocols to make it more secure
- Firewall can be extended to work for all other ports.
- Firewall can be extended to prevent different types of denial of service attacks like “SYN” attacks, Process Table Overflow, Ping of Death.
- This proxy server can be extended as a “caching server” that will cache the pages that are frequently required by the clients in order to reduce the load on the server.

^BIBLIOGRAPHY

[1]V.K. Solanki, K.P. Singh, M. Venkatesan, S. Raghuwanshi, , "*Firewalls Policies Enhancement Strategies Towards Securing Network*", Dept. of CSE, Anna Univ., Chennai, India , in Proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT 2013), 11-12 April 2013.

[2] Tugkan Tuglular, Fevzi Belli, "*Protocol-Based Testing of Firewalls*", Department of Computer Science, Dept. of Comput. Eng., Izmir Inst. of Technol., Izmir, Turkey , 2009 Fourth South-East European Workshop on Formal Methods.

[3] Khaled Salah, Khalid Elbadawi, Member, Raouf Boutaba, "*Performance Modeling and Analysis of Network Firewalls*", Khalifa Univ. of Sci., Sharjah, United Arab Emirates, IEEE transactions on network and service management, vol. 9, no. 1, march 2012

[4] Alex X Lieu, "*Change-Impact Analysis of firewall policies*", ESORICS 2007, LNCA 4734,pp 155-170, Springer-Verlag Berlin Heidelberg 2007.

[5] Ian Mothersole and Martin J. Reed, " *Optimising Rule Order for a Packet Filtering Firewall*", University of Essex, Wivenhoe Park, Colchester, Essex, CO4 3SQ, United Kingdom, Network and Information Systems Security (SAR-SSI), 2011 Conference 18-21 May 2011

[6] Dmifty Rovniagin, Avishai Wool, " *The Geometric Efficient Matching Algorithm For Firewalls*", School of Electrical Engineering, Tel Aviv University, Ramat Aviv 69978, Israel, Dependable and Secure Computing, IEEE Transactions on (Volume:8 , Issue: 1) Jan.-Feb. 2011

[7] Cryptography and Network Security : Principles and Practice by William Stallings

[8] Computer Networks by Andrew S. Tanenbaum

[9] <http://debian-handbook.info/browse/stable/sect.firewall-packet-filtering.html>

[10] <https://publib.boulder.ibm.com/infocenter/iseriess/v5r4/index.jsp?topic=/com.ibm.zos.v5r4.packetff.htm>

[11] <http://securityworld.worldiswelcome.com/packet-filtering-firewall-an-introduction>