IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

Impact Factor: 6.252

Volume 2, Issue 3, April 2022

Layer 7 Packet Filtering Implementation on Actual Kernels

Pratibha Tambewagh[1] and Asmita Jagtap[2]

Lecturer, Bharati Vidyapeeth Institute of Technology, Kharghar, Navi Mumbai, Maharashtra, India[1]

Lecturer, Bharati Vidyapeeth Institute of Technology, Kharghar, Navi Mumbai, Maharashtra, India[2]

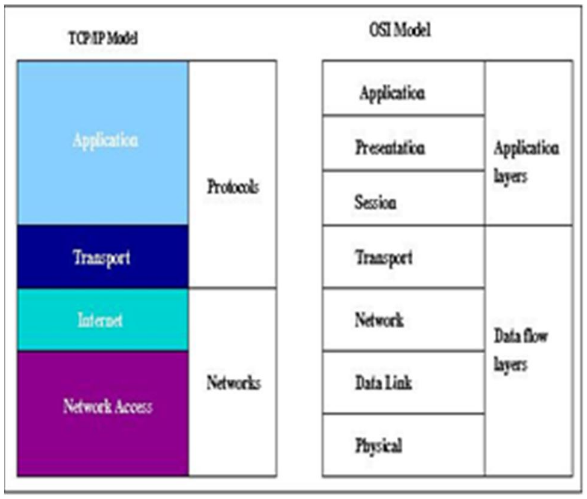
pratibha.tambewagh@gmail.com[1] and smitj24@gmail.com[2]

Abstract: *The use of Application layer packet classifier and optimization of bandwidth towards QoS in Linux using netfilter, iproute2 and layer-7 Filter. As seen in the statistics the huge amount of data flows through the network, so it is necessity to apply packet-filtering rules in order to control the traffic and add firewall rules. Some services are inherently insecure and impossible to secure on individual hosts. Packet filtering tools can help you segment and contain parts of your network to increase security. A packet filtering tools can help you enforce your network security policies by selectively allowing network services. Because a packet filtering tools must examine all inbound/outbound network traffic, it can help you log network activity. We are looking at packet filtering tools like Netfilters and iproute2, who examine the IP packets for filtering and using the queuing disciplines for traffic control.*

Keywords: Application layer packet, HTTP, FTP

I. Introduction

At Layer 7 of the OSI model, we find Application (HTTP, MAIL, DNS. FTP, SSH etc.). As we can see from the following figure (Layer-7 filter), TCP/IP compacted OSI Layers 7, 6, and 5 into one Layer, TCP/IP Layer 4 (Application), which has the same name, but different functionality. Filtering and prioritizing traffic from some applications can be very easy and very hard at the same time. Normally, we would filter/prioritize web traffic by matching TCP packets with source or destination port 80, which is the standard HTTP port. However, web servers can be configured to use any port; so the network traffic filters for particular prioritizations based on source ports, destination ports and the source ip addresses and destination ip addresses won't work for that particular traffic.



Layer-7 Filtering

Filtering traffic belonging to P2P (peer to peer) applications like Kazaa (is a completely distributed peer-to-peer file sharing service), DC++ (is an open source client for the Direct Connect network. Direct Connect allows you to share files over the Internet without restrictions or limits) and many more like Emule (peer-to-peer file sharing clients) etc., as those applications don't use standard ports and, even worse, they can be configured to use other applications' standard ports for communication (e.g. TCP port 80), so to understand and classify the packet the layer 7 protocol is useful.

Layer 7-filter: Is a packet classifier for the Linux kernel that doesn't look up port numbers or Layer 4 protocols, but instead looks up the data in an IP packet and does a regular expression match on it to determine what kind of data it is, mainly what application protocol is being used.

II. Working on Layer-7 Filter

What L7-filter does is provides a way for iptables to match packets based on the application they belong to. The TCP/IP model contains four layers and, before the L7-filter project, netfilter could match data by the first three layers:

Network access layer: iptables -A CHAIN -m mac --mac-source …" Internet: iptables -A CHAIN -s IP_ADDRESS …" Transport: iptables -A CHAIN -p tcp --dport 80 …

At the Network Access layer, netfilter uses -m mac to match packets from or to a MAC address in the network. The Internet layer, we have the IP protocol; netfilter matches packets from or to an IP address, regardless of the transport protocol, port number, or application the packet uses. At the transport layer, we have TCP or UDP, and netfilter can match packets by protocol, and more specifically, by port number within the protocol. Any combination of the three lower layers is permitted, for example

```
iptables –A FORWARD –s 10.100.106.240 –p tcp -–dport 80 –m mac –-macsource 00:01:BC:2D:EF:2A –j DROP"
```

Will drop all packets from the IP address 10.100.106.240 if the source MAC address is 00:01:BC:2D:EF:2A and the packets use the TCP protocol and have the destination TCP port 80.

Layer 7-filter adds a new feature to netfilter by matching packets that belong to an application that is found at the TCP/IP Layer 4. A very important thing is that L7-filter is just another match option for iptables, and so all the rules of the other match options apply in this case. Therefore, you can do all the iptables operations with the packets matched by L7-filter. After adding patch of Layer 7-filter to the kernel, you have -m layer7 --l7proto [http | ftp|snmp|...]

This is the match option we were talking about. In order to match Layer 7 data, netfilter looks deeper into an IP packet than just at its header. However, the actual data contained in the packet doesn't just say "I'm a SNMP packet; filter me!"; so the data is matched against a set of regular expressions that are common to different applications.

This set of regular expressions is probably the most important part of this project, and is called "protocol definitions".The Layer 7-filter contains three important parts:

- A kernel patch, which provides a way for the kernel to look into the IP packets
- iptables patch, which provides the match option for iptables.
- A collection of pattern files that contain the regular expressions for supported protocols (protocol definitions).

2.1 Layer 7 Filtering implementations on actual kernels

To implement Layer 7 filter on actual kernel, we need to patch our kernel with the patch provided by the source found at http://l7-filter.sourceforge.net. To do that, we need the kernel source. The next operation would be to apply the iptables patch, recompile iptables, and install the protocol definitions files.

2.1.1 Applying the Kernel Patch

The first step is to download the kernel source we want from http://www.kernel.org. Next, we need to download L7-filter from at http://l7-filter.sourceforge.net For this project we are using the kernel linux-2.6.18.2 and Layer-7 filter netfilter-layer7-v2.7.tar.gz. After downloading what you need to the /usr/src folder, unzip the L7-filter TAR archive as follows:

```
[root@l7filter src]# tar -xfvz netfilter-layer7-v2.7.tar.gz
[root@l7filter netfilter-layer7-v2.7]# ls -l | awk '{NFS=" "} {print $9}'
[root@l7filter netfilter-layer7-v2.7]#CHANGELOG
```

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, April 2022
Impact Factor: 6.252

```
[root@l7filter netfilter-layer7-v2.7]#for_older_kernels
[root@l7filter netfilter-layer7-v2.7]#iptables-layer7-2.7.patch
[root@l7filter netfilter-layer7-v2.7]#kernel-2.4-layer7-2.7.patch
[root@l7filter netfilter-layer7-v2.7]#kernel-2.6.18-layer7-2.7.patch
[root@l7filter netfilter-layer7-v2.7]#README
[root@l7filter netfilter-layer7-v2.7]#stray_code
```

Next, go to the kernel source root and patch the kernel using the appropriate patch

```
[root@l7filter ~]# cd /usr/src/linux-2.6.18.2
[root@l7filter linux-2.6.18.2]# patch -p1 < ../softwares/netfilter-layer7 2.7/ /kernel-2.6.18-layer7- 2.7.patch
```

It will show the following output.

```
Patching file include/linux/netfilter_ipv4/ip_conntrack.h
patching file include/linux/netfilter_ipv4/ipt_layer7.h
patching file net/ipv4/netfilter/Kconfig
patching file net/ipv4/netfilter/Makefile
patching file net/ipv4/netfilter/ip_conntrack_core.c
patching file net/ipv4/netfilter/ip_conntrack_standalone.c
patching file net/ipv4/netfilter/ipt_layer7.c
patching file net/ipv4/netfilter/regexp/regexp.c
patching file net/ipv4/netfilter/regexp/regexp.h
patching file net/ipv4/netfilter/regexp/regmagic.h
patching file net/ipv4/netfilter/regexp/regsub.c
```

Now compile the kernel using, run make config, make menuconfig, or make Xconfig. You need to enable the following options:

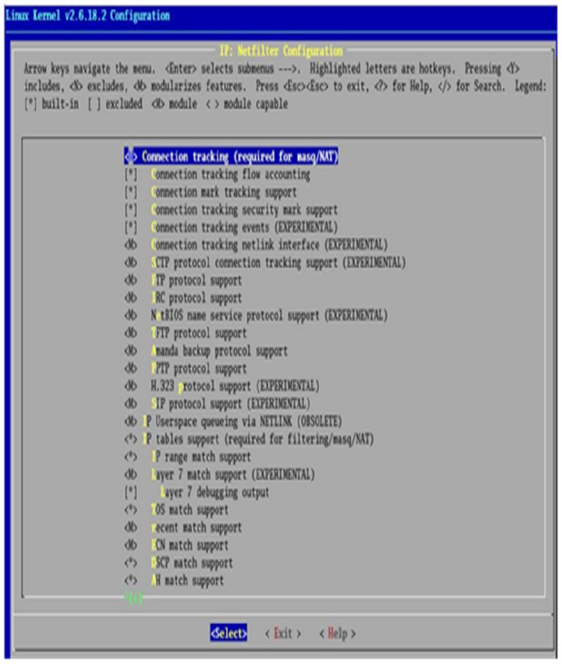
Code maturity level options | Prompt for development and/or incomplete code/drivers.

Netfilter (Device Drivers | Networking support | Networking Options | Network packet filtering).

Connection tracking (Network packet filtering | IP: Netfilter Configuration Connection tracking).

Connection tracking flow accounting and IP tables support. Layer 7 match support.

Following modules should be selected during kernel compilation. That is Layer 7 match support.



```
Linux Kernel v2.6.18.2 Configuration

IP: Netfilter Configuration
Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.  Pressing <Y>
includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend:
[*] built-in  [ ] excluded  <M> module  < > module capable

<M> Connection tracking (required for masq/NAT)
[*]   Connection tracking flow accounting
[*]   Connection mark tracking support
[*]   Connection tracking security mark support
[*]   Connection tracking events (EXPERIMENTAL)
<M>   Connection tracking netlink interface (EXPERIMENTAL)
<M>   SCTP protocol connection tracking support (EXPERIMENTAL)
<M>   FTP protocol support
<M>   IRC protocol support
<M>   NetBIOS name service protocol support (EXPERIMENTAL)
<M>   TFTP protocol support
<M>   Amanda backup protocol support
<M>   PPTP protocol support
<M>   H.323 protocol support (EXPERIMENTAL)
<M>   SIP protocol support (EXPERIMENTAL)
<M> IP Userspace queueing via NETLINK (OBSOLETE)
<*> IP tables support (required for filtering/masq/NAT)
<*>   IP range match support
<M>   Layer 7 match support (EXPERIMENTAL)
[*]     Layer 7 debugging output
<*>   TOS match support
<M>   recent match support
<M>   ECN match support
<*>   DSCP match support
<*>   AH match support
v(+)

<Select>   < Exit >   < Help >
```

Figure: Kernel compilation

2.1.2 Applying the iptable patch

To apply the iptables patch, we need the iptables sources from http://www.netfilter.org. Go to the iptables source root and patch it with the patch provided by the L7-filter project.

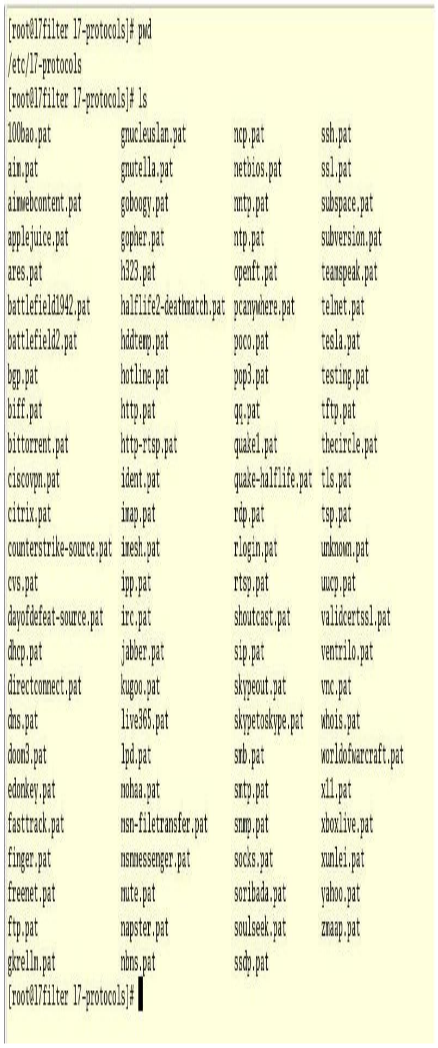
```
[root@l7filter iptables-1.3.7]# patch -p1 < ../softwares/netfilter-layer7-v2.7/iptables-layer7-2.7.patch
```

It will show the following output.

```
patching file extensions/.layer7-test patching file extensions/libipt_layer7.c
```

2.1.3 Protocol Definitions

First, we need to download the protocol definitions archive from the L7-filter project page at sourceforge,http://prdownloads.sourceforge.net/l7-filter/l7-protocols-2007-01-14.tar.gz download. And we need to copy the pattern files (.pat) from the archive to the /etc/l7-protocols folder. Following are the application patterns supported by L7 filter



```
[root@l7filter l7-protocols]# pwd
/etc/l7-protocols
[root@l7filter l7-protocols]# ls
100bao.pat               gnucleuslan.pat          ncp.pat           ssh.pat
aim.pat                  gnutella.pat             netbios.pat       ssl.pat
aimwebcontent.pat        goboogy.pat              nntp.pat          subspace.pat
applejuice.pat           gopher.pat               ntp.pat           subversion.pat
ares.pat                 h323.pat                 openft.pat        teamspeak.pat
battlefield1942.pat      halflife2-deathmatch.pat pcanywhere.pat    telnet.pat
battlefield2.pat         hddtemp.pat              poco.pat          tesla.pat
bgp.pat                  hotline.pat              pop3.pat          testing.pat
biff.pat                 http.pat                 qq.pat            tftp.pat
bittorrent.pat           http-rtsp.pat            quake1.pat        thecircle.pat
ciscovpn.pat             ident.pat                quake-halflife.pat tls.pat
citrix.pat               imap.pat                 rdp.pat           tsp.pat
counterstrike-source.pat imesh.pat                rlogin.pat        unknown.pat
cvs.pat                  ipp.pat                  rtsp.pat          uucp.pat
dayofdefeat-source.pat   irc.pat                  shoutcast.pat     validcertssl.pat
dhcp.pat                 jabber.pat               sip.pat           ventrilo.pat
directconnect.pat        kugoo.pat                skypeout.pat      vnc.pat
dns.pat                  live365.pat              skypetoskype.pat  whois.pat
doom3.pat                lpd.pat                  smb.pat           worldofwarcraft.pat
edonkey.pat              mohaa.pat                smtp.pat          x11.pat
fasttrack.pat            msn-filetransfer.pat     snmp.pat          xboxlive.pat
finger.pat               msnmessenger.pat         socks.pat         xunlei.pat
freenet.pat              mute.pat                 soribada.pat      yahoo.pat
ftp.pat                  napster.pat              soulseek.pat      zmaap.pat
gkrellm.pat              nbns.pat                 ssdp.pat
[root@l7filter l7-protocols]#
```

Malware

This category includes worms, viruses, and anything else that uses the network to bother us. It doesn't appear that there is much demand for this functionality, but in case it interests you, this is a proof-of-concept.

Name	Description
code_red	Code Red - a worm that attacks Microsoft IIS web servers
nimda	Nimda - a worm that attacks Microsoft IIS web servers, and MORE!

III. How to write Layer 7-filter Patterns

To write basic format in the following way:

1. The name of the protocol on one line
2. A regular expression defining the protocol on the next line (see regular expressions below)

The name of the file must match the name of the protocol. (If the protocol is "ftp", the file must be "ftp.pat".)

3.1 Meta-data

In the Pattern files, top four lines should look like this:

# <Protocol name and some concise detail about the protocol>
# Pattern attributes: [attribute word]*
# Protocol groups: [group name]*
# Wiki: [link]*

Pattern attributes" give information about how good the pattern is on various scales. Attribute words can be any of undermatch, overmatch, superset, subset, great, good, ok, marginal, poor, veryfast, fast, nosofast, or slow. Any number of these may be used. They are defined on the protocol pages. "Protocol groups" are supposed to give frontends a way to group similar protocols. Group names can be whatever you like, but should match existing names if possible. Any number may be used.

3.2 Regular Expressions

The kernel and userspace versions of l7-filter use different regular expressions libraries. They use generally the same syntax, but have some differences.

General information

Because patterns frequently need to use non-printable characters, both versions of l7-filter add perl- style hex matching on top of their stock libraries. This uses \xHH notation, so to match a tab, use "x09". Note that regexp control characters are still control characters even when written in hex:

```
\x24 == $
\x29 == )    \x28 == ( \x2a
 \x2b == +         == * \x2e
 \x3f == ?         == . \x5b
 \x5c == \         == [ \x5d
 \x5e == ^ \x      == |== ]
7c == |\x7b == { (only a control character for the userspace) \x7d == } (only a control character for the userspace)
```

3.3 What the Classifier Sees

If you have set up your iptables rules correctly, the classifier sees the data going in both directions in the order that it passes through the computer. For instance, in FTP, the first thing the filter sees is "221 server ready", then "USER bob", then "331 send password", then "PASS frogbeard", and so on. l7-filter can match across packets. For instance, you could match FTP with "220.*user.*331".

IV. What Makes a Good Pattern

There are two general guidelines:

1) A pattern must be neither too specific nor not specific enough.

Example 1: The pattern "bear" for Bearshare is not specific enough.

This pattern could match a wide variety of non-Bearshare connections. For instance, an HTTP request for http://bear.com would be matched.

Example2: "220 .*ftp.*(\[.*\]|\(.*\))" for FTP is too specific.
Not all servers send () s or []s after their 220. In fact, servers are not even required to send the string "ftp" at any time, but the vast majority do. Good judgment and testing are necessary for instances such as this.
2) It should use a minimum of processing power. If it's possible to reduce the number of instances of *, + and | in your pattern, you should do so. Use the performance testing program included in the patterns package.
3) It should complete its match on the earliest packet possible.

The FTP pattern could be "^220[\x09-\x0d -~]*\x0d\x0aUSER[\x09-\x0d -~] *\x0d\x0a331", but that won't match until the third data packet. Instead, we use "^220[\x09-\x0d -~]*ftp", which matches on the first data packet.

4.1 Recommended Procedure for Writing Patterns

1. Find and read the spec for the protocol you wish to match. If it's an Internet standard, RFCs are a good place to start, although not all standards are RFCs. If it is a proprietary protocol, it is likely that someone has written a reverse-engineered spec for it. Do a general web search to find it. Skipping this step is a good way to write patterns that are overly specific!
2. Use something like Wireshark (formerly known as Ethereal) to watch packets of this protocol go by in a typical session of its use. (If you failed to find a spec for your protocol, but Wireshark can parse it, reading the Wireshark source code may also be worth your time.)
3. Write a pattern that will reliably match one of the first few packets that are sent in your protocol. Test it. Test its performance

4.2 Testing the Layer-7 filter Installation

First, we might want to see if our module is in place. We can do that using the modinfo command:

```
[root@l7filter softwares]# modinfo ipt_layer7
filename:       /lib/modules/2.6.18.2/kernel/net/ipv4/netfilter/ipt_layer7.ko
author:   Matthew Strait <quadong@users.sf.net>, Ethan Sommer <sommere@users.sf.net>
license:  GPL
description:    iptables application layer match module
version:  2.0
parmtype:       maxdatalen:int
parm:     maxdatalen:maximum bytes of data looked at by l7-filter
vermagic:       2.6.18.2 mod_unload 686 REGPARM 4KSTACKS gcc-3.4
depends:x_tables,ip_conntrack
srcversion:     C5460962D1CE10F665D072A
```

The output shows that we have a module called ipt_layer7 and some information about it, such as filename, author, license, description, version, and other module dependencies.
Next, we will try to load the module using the modprobe command:

```
[root@l7filter softwares]# modprobe ipt_layer7 [root@l7filter softwares]# lsmod
                         Module        Size        Used by

                         ipt_layer7    12932       0
```

The module loaded into the kernel, its size, and the number of processes it is used by (in our case 0), because we didn't used it yet. When downloading the files, we should see that all packets are match and we will insert an accounting rule in the OUTPUT chain to match all the outgoing HTTP traffic.

```
[root@l7filter ~]# /usr/local/sbin/iptables -A OUTPUT -m layer7 --l7proto http
 [root@l7filter ~]# /usr/local/sbin/iptables -L OUTPUT -n -v
 Chain OUTPUT (policy ACCEPT 31 packets, 3164 bytes)
 pkts bytes target    prot opt   in   out   source        destination
 0   0       0        --         *    *     0.0.0.0/0     0.0.0.0/0          LAYER7 l7proto http
```

```
[root@l7filter ~]# wget http://127.0.0.1/Qos.ps --17:23:25-- http://127.0.0.1/Qos.ps
=> `Qos.ps'
Connecting to 127.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK Length: 24,803,838 (24M) [application/postscript]
100%[=========================================>] 24,803,838 53.65M/s 17:23:25 (53.64 MB/s) -
`Qos.ps' saved [24803838/24803838]

    [root@l7filter ~]# /usr/local/sbin/iptables -L OUTPUT -n -v
    Chain OUTPUT (policy ACCEPT 31 packets, 3164 bytes)
     pkts       bytes
target                 prot opt in    Out   Source        destination
0   0       0          --   *         *     0.0.0.0/0     0.0.0.0/0          LAYER7 l7proto http

                        [root@l7filter ~]#
                      modprobe ip_conntrack
                      [root@l7filter ~]# /usr/local/sbin/iptables -L OUTPUT -n -v
                      Chain OUTPUT (policy ACCEPT 2402 packets, 25M bytes)
                       pkts   prot
                      bytes   opt
                      target   in   out  source  destination
                                                                           LAYER7
                       2290   0  --                                        l7proto
                       25M     *     *  0.0.0.0/00.0.0.0/0                 http
```

Well, as you can see, it worked. Now we have a Linux router with application layer filtering capabilities.

V. Layer7-Filter Applications

We can use L7-filter with any iptables option; L7-filter provides just another match option. L7-filter might match packets belonging to other applications than the one you want.

5.1 Filtering Application Data

Blocking unwanted applications data that passes through your router is one things that you can do with L7-filter. Traffic from different applications might look similar; so you might experience problems when dropping data based on the L7-filter match. For example, if you drop packets that belong to eDonkey, there might be some other protocols that will experience problems. The eDonkey pattern matches about 1% of other streams with random data. If you still want to use L7-filter for blocking several applications passing through your Linux router, it can be done as follows:

```
[root@l7filter ~]# iptables -A FORWARD -m layer7 --l7proto edonkey -j DROP
```

For port number and ip addresses.

```
[root@l7filter ~]# iptables -A FORWARD -m layer7 --l7proto edonkey –d 10.100.106.244 -j DROP
```

Finally, L7-filter is used for small to medium networks that need bandwidth optimization. The advantage of L7-filter over the specialized hardware solutions is, of course, the cost. Use L7-filter if it doesn't affect the network performance and doesn't overload the router's CPU.

The decision whether to use L7-filter must be based on the machine performance (mainly CPU speed) and the type of traffic passing through it . L7-filter is recommended to be used for marking packets in order to queue them.

References

[1]. Lucian Gheorghe “Designing and Implementing Linux Firewalls and QoS using netfilter, iproute2, NAT, and LFilter”
[2]. Packt Publishing, October 2006

[3]. Application Layer Packet Classifier for Linux website “http://l7filter.sourceforge.net/”

[4]. Netfilter, firewalling, nat and packet mangling for linux website http://www.netfilter.org

[5]. Nigel Kukard “Bandwidth Management and Optimization” International Network INASP, Open source Bandwidth Solutions March 2006

[6]. Lukas Kencl, Christian Schwarzer, “Traffic Adaptive Packet Filtering of Denial of Service Attacks” Intel Research laboratories, World of Wireless, Mobile and Multimedia Networks,2006. WoW MoM 2006.

[7]. J. McCann and Satish Chandra, “Packet Types: Abstract Specification of Network Protocol Messages” Bell Laboratories, ACM SIGCOMM Computer Communication Review Volume 30 , Issue 4 October 2000

[8]. Jeffrey C. Mogul, “ The Packet Filter An Efficient Mechanism for Userlevel Network Code ”Digital Equipment Corporation Western Research Laboratory, ACM Operating Systems Review, SIGOPS

[9]. Holger Dreger, Anja Feldmann, Michael Mai, Vern Paxson, Robin Sommer “Dynamic Application Layer Protocol Analysis for Network Intrusion Detection” USENIX Security

[10]. Pankaj Gupta ,Nick McKeown “Packet Classification on Multiple Fields”, Proc. Sigcomm, Computer Communication Review, vol. 29, no. 4, pp 14760,September 1999, Harvard University.

[11]. Florin Baboescu George Varghese “Scalable Packet Classification” , University of California, San Diego Proceedings of ACM Sigcomm, pages 199210, August, 2001.