ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

Performance Improvement of a Packet Filter By Filtering Compressed Packet

Archita Dad[1], Anil Saroliya[2]
Student, Computer Science, Amity University, Rajasthan, India [1]
Assistant Professor, Computer Science, Amity University, Rajasthan, India [2]

Abstract: Internet is used as an extensive source for communication. It has also growths the unlawful activities by terrorist and criminals to communicate information. For crime detection and prevention, law enforcement agencies need to keep up with the rising trends in these areas and needs a tool to monitor network traffic. This tool is not a need of only law enforcement agencies but also commercial sector so that companies can prevent their valuable data from falling into hands of their competitor. Now day's these tools are provided with network devices and also available in market such as PickPacket, PKTD, JPCAP, NetXMS, etc. Data can be filtered at 3 levels in these tools by network parameters, application specific and content specific Filter. To speed up the transaction and for security reasons data is used in compressed form on internet. In content specific level it is difficult to apply sting searching algorithm on compressed data. This paper presents a solution to decompress HTTP data on network.

Keywords: Pick-Packet, HTTP, TELNET, FTP, Compressed Data, gzip**.**

I. INTRODUCTION

In last few decades internet has an exponential growth. Large volume of data can be exchanged on internet. This has resulted in an ever increasing need for effective tools that can monitor the network. Basic goal of network monitoring is to read packets from network and analyze its content.

Introduction part of this paper contains the basic idea about Network monitoring tool and pick-packet. Second point describes the architecture and levels of pick-packet. It also describes the working of Pick-Packet. Third section of this paper describes the application protocol supported by Pick-Packet specially HTTP protocol. After that need of compressed data and processed of filtering compressed data is discussed. In fifth section how we can filter compressed data on fly is shows and what are the modifications needed after filtering data compression on fly is shown in last section. After that conclusion of this paper is describe.

Network monitoring tools are also known as Sniffer. Sniffers have used in law enforcement agencies for crime detection and prevention and in unlawful activities to break the computer. Generally sniffers work by putting the Network Interface Card into promiscuous mode. In this mode the Ethernet card listens to "all the traffic which is coming in". If the Network Interface Card is not in promiscuous mode, it ignores all traffic which is not intended for it. Filtering can be done in two modes, on-line filtering and off-line filtering. On-line filtering is implemented in kernel while capturing the traffic. Off-line filtering is done after the captured data is stored on disk. However sniffers can be rendered useless through the use of encryption mechanisms. Several tools exist that can monitor network traffic. Usually such tools put the network card of a computer into the "promiscuous mode". This enables the computer to listen to the entire traffic on that subsection of the network. There can be an additional level of filtering of these packets based on the IP related header data present in the packet. Usually such filtering specifies simple criteria for the IP addresses and ports present in the packet. Filtered packets are written on to the disk. And after that offline analysis is done on these packets to gather the required information from these packets. [1]

Pick-Packet is a monitoring tool that can filter packets across the network layer and application layer of the OSI network stack for selected applications. Criteria for filtering can be specified in Pick-Packet Configuration File Generator for network layer and application layer for applications like

TELNET, SMTP, HTTP, FTP etc. It also supports real-time searching for text string in application and packet content. Pick-Packet filter the packet according to specified criteria in configuration file and store them to some for further processing.

A special provision has been made in the tool for two modes of capturing packets depending on the amount of granularity with which data has to be captured. These are the "PEN" mode and the "FULL" mode of operations. In the first mode it is only established that a packet corresponding to a particular criterion specified by the user was encountered and minimal information required for further detailed investigation is captured. In the second mode the data of

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

such a packet is also captured. Judiciously using these features can help protect the privacy of innocent users.
The packets dumped to the disk are analyzed in the offline mode. Post dump analysis makes available to the investigator separate files for different connections. The tool provides a summary of all the connections and also provides an interface to view recorded traffic. A GUI for generating the input rules to the filter is also provided.[2,3]

II. PACKET FILTERING BY PICK-PACKET

Pick-Packet can be logically viewed as aggregate of four components working in pipeline, ideally deployed on four different machines. These components are – the *Pick-Packet Configuration File Generator*, the *Pick-Packet Filter*, the *Pick-Packet Post Processor* and the *Pick-Packet Data Viewer* GUI. An architectural view of Pick-Packet is shown in Figure 1.

A. Pick-Packet Configuration File Generator

The Pick-Packet Configuration File Generator is a java based graphical user interface (GUI) that generates the configuration file that is input to the Pick-Packet Filter. The user can specify different filtering criteria for filtering the data which is subsequently written to configuration file in a format that is understood by the filter.

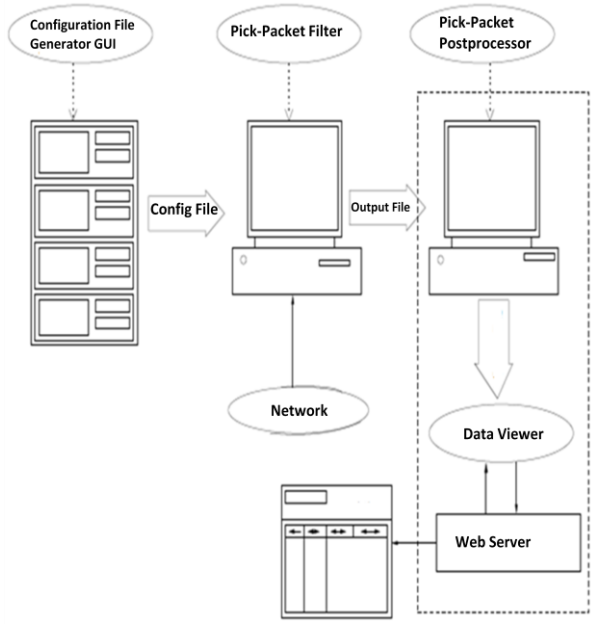

Fig 1: Pick-Packet Architecture [1]

B. Pick-Packet Packet Filter

The Pick-Packet packet filter takes the configuration file as input. It reads packet from the network and stores those packets which match the criteria specified in configuration file. Filtering is done at different levels based on criteria like IP address, port number and application layer information.

C. Pick-Packet Post Processor

The Pick-Packet Post Processor processes the packet stored on the disk, and retrieves the meta-information from them and creates a directory structure which is used by the Data Viewer.

D. Pick-Packet Data Viewer

The Pick-Packet Data Viewer is a web based GUI. It takes the directory created by the post-processor as input and displays the data in an interactive manner.[4]
Pick-Packet Packet Filter takes packet from network and filters them on basically three levels.

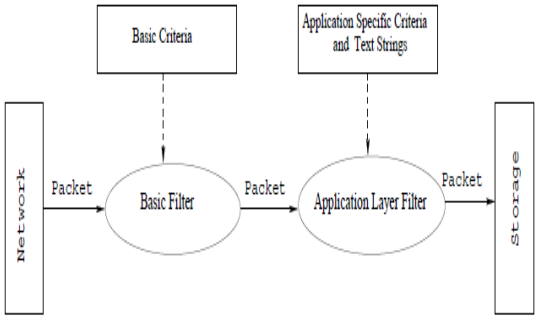

Fig 2: Levels of Filtering [3]

1. Filtering based on network parameters (IP address, Port No., etc)
2. Filtering based on application layer protocol specific criteria (user name, email id, etc.)
3. Filtering based on content present in application pay load (text string search, etc.)[5]

Since it would be convenient to have different filters for different application layer protocol based filters, the combined second and third level filtering can be split into several application specific filters one for each application. If this model of filtering is chosen a demultipelexer is required between the first level filter and application specific filters so that each application gets only relevant packets. The demultiplexer uses its own set of criteria for demultiplexing packets as shown in fig 3.

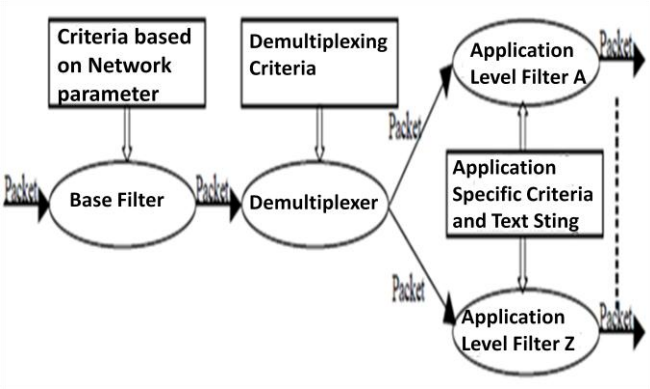

Fig 3: Demultiplexing packets for filtering

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

The purpose of Pick-Packet, like the filters discussed above is to monitor network traffic and to copy only selected packets for further analysis. However, the scope and complexity of criteria that can be specified for selecting packets is greatly increased.

The criteria for selecting packets can be specified at several layers of the protocol stack. Thus there can be criteria for the Network Layer – IP addresses, Transport Layer – Port numbers and Application Layer – Application dependent such as file names, email ids, URLs, text string searches etc. The filtering component of this tool does not inject any packets onto the network. Once the packets have been selected based on these criteria they are dumped to permanent storage. [1, 5, 6, 7]

Fig 4 shows the basic design of the Pick-Packet Filter. The filter takes as input a configuration file that contains packet capturing parameters at the various levels of filtering. First set of parameters are the IP addresses and ports. These parameters are used by the in-kernel filtering module – the Basic Filter and the Demultiplexer. The second sets of parameters are required by the Application Level Filters. These parameters are application specific and include user names, URLs, the text string to be searched etc. The third sets of parameters are used by an Output File Manager that controls the format of the dumped packets and the size of the output file etc. It is the task of the Demultiplexer to route packets. Sequencing of packets may have to be determined to perform searches on text split between packets.

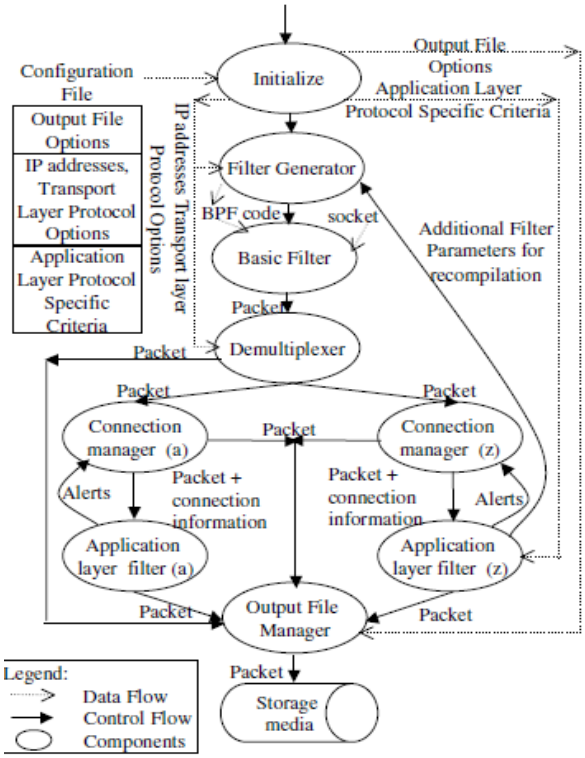

Fig 4: Basic design of Pick-Packet Filter [7]

The Demultiplexer, therefore, routes packets to a Connection Manager that performs this task. The Connection Manager passes packets to the Application Level Filter. There are several considerations that go into designing the connection manager. First the connection manager need not determine the sequencing of packets for all connections. Rather, it should determine sequencing for only those connections that an application layer filter is interested in. Communication between the application layer filter and the connection manager to indicate such interest is provided by means of alerts. A second consideration pertains to the level at which history data is remembered for an application. A cursory design would store remembered data at the application layer level. Searching for this data is done based on the four tuples (source IP, destination IP, source port and the destination port). [8,9]

However this four tuple is also examined by the demultiplexer. States dependent on this four tuple are also maintained by the connection manager. Therefore it is best to pass the data that the application wishes to associate with a connection to the connection manager and subsequently to the demultiplexer. Alerts incorporate this mechanism also. [4]

III. APPLICATION PROTOCOL SUPPORTED BY PICK-PACKET

Pick-Packet filter support many application level filters such as TELNET, HTTP, FTP, SMTP and RADIUS. [5, 6, 7] For each application protocol a filter is used which is work on the criteria specified by user. Header of a packet is checked by a filter on specified criteria. If it matched packet is dumped on disk.

Each application level filter is capable of searching for parameters relevant to the application also. Thus the SMTP filter can search for senders, receivers and addresses. Similarly the FTP filter can search for user names, file names and the HTTP filter can search for URLs.

```
GET http://www.tiggerwigger.com/ HTTP/1.0
Proxy-Connection: Keep-Alive
User-Agent: Mozilla/5.0 [en] (X11; I; Linux 2.2.3 i686)
Host: www.tiggerwigger.com
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */
*
```

Fig 5: HTTP Request Header

For example an HTTP request header and HTTP response header contains basic information about the host, URL, http version, browser and many other field that are used by the HTTP Application Level Filter to filter an HTTP packet.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

Additional features of HTTP 1.1 that have been addressed by the HTTP Filter are persistent connections, chunked transfer encoding and the "HOST:" header. Persistent connection also allows pipelining of requests. Client can send requests to the server without waiting for a response. This directly impacts the HTTP Filter as a single packet can content multiple requests. Chunked Transfer Encoding has a direct bearing on the HTTP Filter.

If a response has to be sent before its total length is known the simple chunked transfer-encoding can be used. This breaks the complete response into smaller chunks and sends them in series. Such a response can be identified as it contains the "Transfer-Encoding: chunked" header. A chunked message body contains a series of chunks, followed by a line with "0" (zero), followed by optional footers, and a blank line. Sample headers shown in figure 5 and 6:

```
HTTP/1.0 200 OK
Date: Fri, 13 Nov 2009 06:57:43 GMT
Content-Location: http://locutus.tiggerwigger.com/index.html
Etag: "07db14afa76be1:1074"
Last-Modified: Thu, 05 Nov 2009 20:01:38 GMT
Content-Length: 7931
Content-Type: text/html
Server: Microsoft-IIS/4.0
Age: 922
Proxy-Connection: close
```

Fig 6: HTTP Response Header

In HTTP filter a provision have been made for specifying host names, paths, and text string that will be monitored in HTTP connection. Once a host name and path has matched in some packet of a HTTP connection the message body of HTTP request and response as well as URI is searched for a match of a specified text string. If all the criteria specified by user match for connection request packet are dumped to disk.

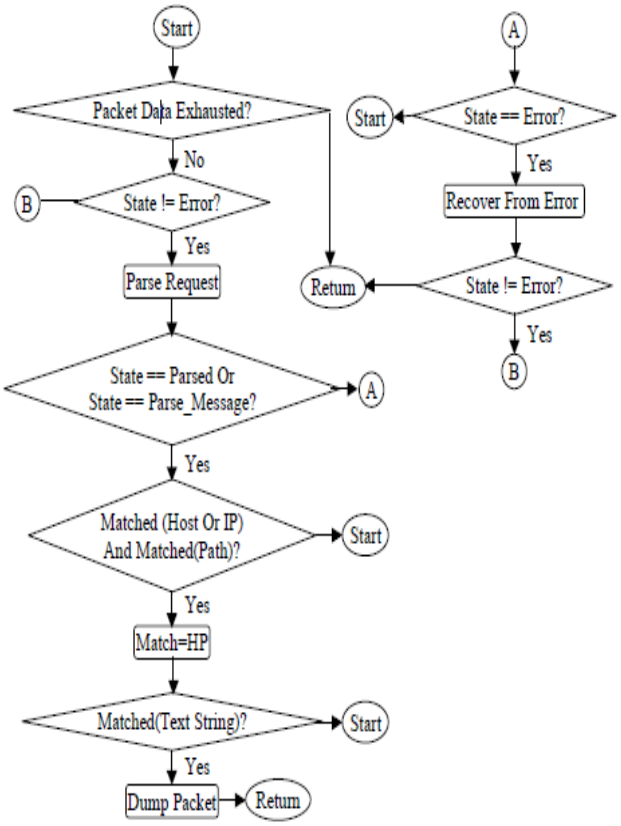

Fig 7: Filtering a HTTP Response Packet [3]

HTTP filter has a structure that is allocated for each connection. This structure holds the information pertaining to that connection. Response and request structure are important member of this structure. Handling of HTTP request and packet by HTTP Filter is shown in figure 5 and figure 6 the flowchart for respectively. [5]

The basic idea behind the flowchart is to parse the packet in a loop till packet data is exhausted. The parser for the request consumes the packet data and returns after setting states for the request structure. Data may be left in the packet after parsing because of pipelining or error. After the parser returns further processing is necessary if parsing has either parsed an entire request or has retrieved partial content of request. The parser may be able to retrieve partial content in cases where the message body of request is split across packets. Under these conditions, the data retrieved from the packet by the parser is checked for match of user supplied criteria. If the criteria match the connection can be dumped otherwise, if the entire packet data has been exhausted, the packet can be put into a list of history packets. [5]

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

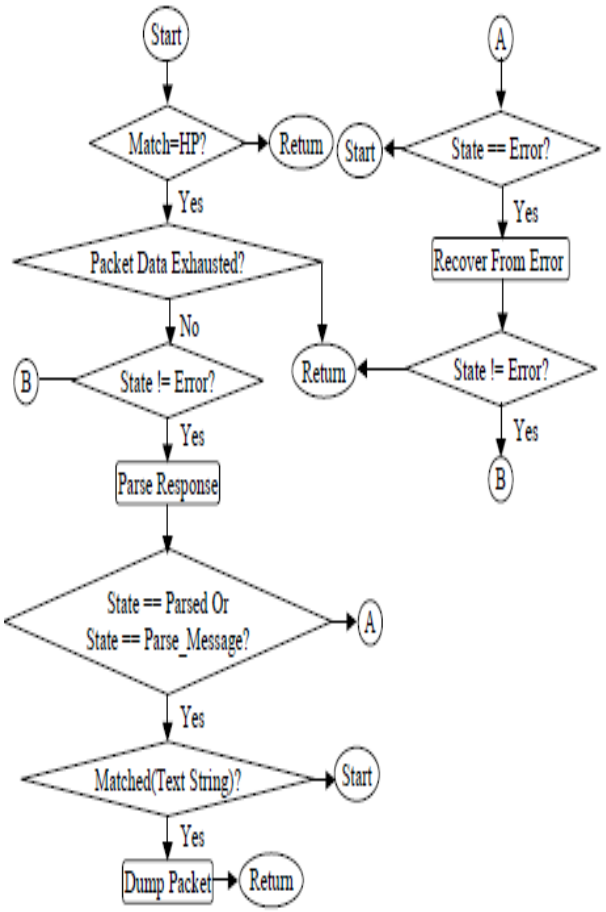

Fig 8: Filtering a HTTP Response Packet [3]

IV. APPLICATION PROTOCOL SUPPORTED BY PICK-PACKET

HTTP filtering as shown in the above example is done on normal text or image format as mentioned by the request header's accept field and response header's content type field. This type of filtering is done before saving data on disk. It is also known as On-line filtering or data on fly filtering.

```
GET / HTTP/1.1
Host: www.cricinfo.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.5)
    Gecko/20031007 Firebird/0.7
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,
    text/plain;q=0.8,video/x-mng,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

Fig 9: HTTP Request Header for Compressed Data

But normal text or image takes more time to download a web page. So to speed up the transactions and full utilization of available bandwidth compression is now being used on web. The compressed data at server can be generated in two types, dynamically and pre-compressed. Browsers and servers have a brief conversation before actual transfer of compressed data for example HTTP compression is introduce to improve web performance by having the server send compressed files to client and having the browser uncompress before displaying. HTTP compressions accept standard gzip and deflate encoding algorithms to compress CSS, XHTML and JavaScript to speed up web page downloads and save bandwidth.

```
HTTP/1.1 200 OK
Date: Thu, 04 Dec 2003 16:15:12 GMT
Server: Apache/2.0
Vary: Accept-Encoding
Content-Encoding: gzip
Cache-Control: max-age=300
Expires: Thu, 04 Dec 2003 16:20:12 GMT
Content-Length: 533
Content-Type: text/html; charset=ISO-8859-1
```

Fig 10: HTTP Response Header for Compressed Data

Before compressed data is transferred conversation is done using HTTP headers. A compression-aware browser's HTTP request message informs the server that it prefers to receive compressed data.

When web server receives request header with *Accept-Encoding: gzip*, it delivers the requested document with the encoding accepted by client. Client downloads the compressed file, displays the pages after decompression. Content filtering will works only on decompressed data so to apply text string searching algorithm ones need to store all the packets into a buffer until the whole file is captured, then decompress the file and apply string searching algorithm. This method does not support data on the fly filtering.

V. COMPRESSED DATA FILTERING

To apply content filtering (string search algorithm) on compressed data packet ones need to store these packets on disk and decompressed the data because it is difficult to apply on the fly string searching algorithm on the compressed data.[10] This violates the entire goal of packet filtering (Pick-Packet).

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

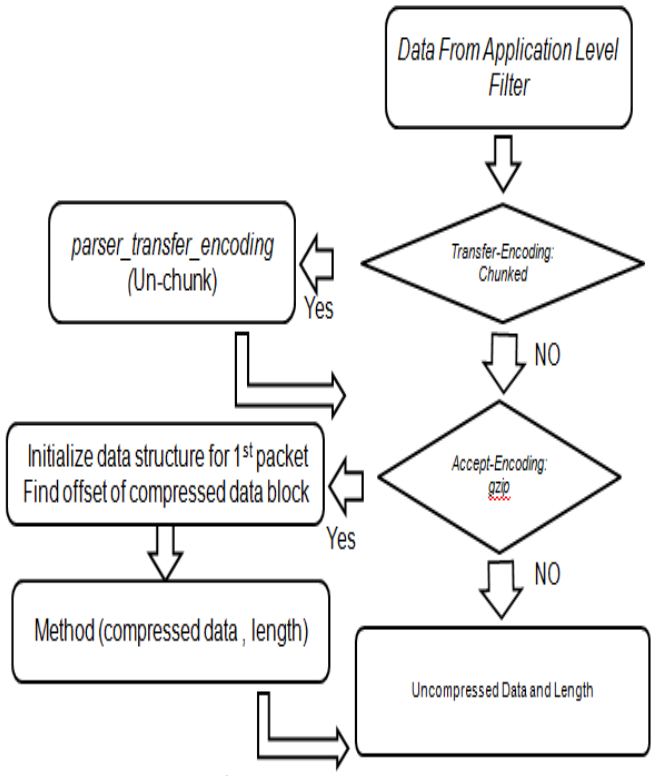

Fig 11: Content filtering on compressed data

Another efficient solution for data on fly filtering is provided in this paper. When HTTP application layer filter parse the response header, it checks whether there is any *Transfer-Encoding: Chunked* is present. It shows that file is transferred in no. of packets. If it present Pick-Packet calls *parser_transfer_encoding* to un-chunk the chunked data. Otherwise directly comes to *parse_content_encode.* In this *Accept-Encoding: gzip* field is checked. If it is found HTTP connections data structure is initialized with default values for first packet and checks for gzip file header values one after another and find offset of compressed data block. For other packets a method is called with compressed content as a parameter. This method takes compressed data and its length as input parameters and return equivalent uncompressed data and its length. It remembers the data structure to decompress next consecutive packet of a file. Decompression on fly succeeds only if packets received in order. It fails to decompress a packet if packets before it have been lost.

On successful decompression of compressed HTTP data, pick-packet sends uncompressed data to string search methods of application level filtering to match user specified keywords. If match occurs, then the packet is stored to disk for future analysis.

VI. UPGRADING OF POST-PROCESSOR

Post-Processor needs to handle compressed HTTP data. Data Viewer needs some meta-information to show the connection data along with criteria because of which connection is selected. If those criteria are keyword, then it displays that keyword and frequency of that keyword in the connection. In order to get keyword and to find its frequency in compressed data, Post-Processor has to uncompress the packet data and apply string search algorithm. Finally after processing of compressed HTTP data, Pick-Packet stores it in compressed form only. In web based Data Viewer, when we request that content, browser will get that from the disk and uncompress it and displays it to the user.

VII. CONCLUSION

Network monitoring tools are essential need of today's word. In these days to speed up the transaction data are sent in compressed format. Some Network monitoring tools can't apply the content filter on compressed data before storing it on disk like pick-packet.

This paper takes pick-packet as example of network monitoring tool and tried to implement text string searching on HTTP compressed data on fly. But the basic condition for filtering the data on fly is that packets must receive in order. Same method may be also applicable on other application layer protocol.

REFERENCES

[1] N. Kapoor. "Design and Implementation of a Network Monitoring Tool". Master's thesis, Dept. of Computer Science and Engg., IIT Kanpur, Apr 2002.

[2] Hal Abelson, Ken Ledeen, Chris Lewis (2009). "Just Deliver the Packets, in: "Essays on Deep Packet Inspection", Ottawa". Office of the Privacy Commissioner of Canada. Retrieved 2010-01-08.

[3] Spy-Gear Business to Be Sold - Amesys to Sell Business That Provided Surveillance Technology Used by Gadhafi, the Wall Street Journal, German edition, friday, march the 9th of 2012.

[4] "The Book of PF: A No-Nonsense Guide to the OpenBSD Firewall" by Peter N. M. Hansteen. No Starch Press. Second edition 2010. ISBN 978-1-59327-274-6.

[5] B. Pande. "The Network Monitoring Tool - Pickpacket: Filtering FTP and HTTP Packets". Master's thesis, Dept. of Computer Science and Engg., IIT Kanpur, Sept 2002.

[6] A. Prashant. "Pickpacket: Design and Implementation of the HTTP Postprocessor and MIME Parse Decoder". Technical report, Dept. of Computer Science and Engg., IIT Kanpur, Apr 2003.

[7] S. K. Jain. "Implementation of RADIUS Support in Pickpacket". Master's thesis, Dept. of Computer Science and Engg., IIT Kanpur, May 2003.

[8] S. McCanne and V. Jacobson. "The BSD Packet Filter: A New Architecture for User-level Packet Capture". In Proc.of USENIX Winter Conf., pages 259–269, Jan 1993.

[9] R. Boyer and J. Moore. "A Fast String Searching Algorithm". In *Comm. ACM 20*, pages 762–772, 1977.

[10] V. Jacobson, C. Leres, and S. McCanne. "pcap - PacketCapture Library", 2001. Unix man page.