Study and Investigation of Convention Network analyzer and payload data to Remake Correspondence between two Nodes in a System

Preeti Raj Verma
M.Tech Scholer
Dr. A.P.J Abdul Kalam Technical University Lucknow,India
rajpreeti18@gmsil.com

Navpreet Singh
Chief Engineer, Computer Centre,
IIT Kanpur,India
navi@iitk.ac.in

Abstract: In this paper, we think about and dissect parcel payload data to remake correspondence between two Node in a system. Information going as bundles or packets is caught utilizing TCPDUMP. By taking a gander at port number, convention of information transmission can be judged effectively. Likewise if information is going in plain content arrangement, then genuine data going between hosts can be effectively retrieved. This paper gives a technique to concentrate packet header and payload data and utilize it to reproduce the correspondence. On the off chance that the information is moved in plain content organization, then the total content can be retrieved. Currently TELNET, FTP, SSH, HTTPS, HTTP and SMTP applications have been considered. Comparative approach can be reached out for different applications like record sharing and VoIP as well. This data can be extremely valuable for security organizations which can utilize this to reproduce the correspondence occurring from any PC on the system. The message being transported can likewise be recovered in the event of plane content correspondence**.**

Keywords: TCPDUMP, HTTPS, FTP, SMTP**,** SSH, SFTP.

I. INTRODUCTION

In this Paper, we think about and break down System Conventions Packet header and payload data we make a parcel which catch the at present running system movement and we play out some particular investigation to improve utilization of this checking tool. The first favorable position of our innovation is the capacity to create estimations progressively and it is the electronic application which effectively interfaces rapid system and begin observing as needs be, second, the device can be effortlessly reached out to consider a few sorts of system conventions Packet header and payload analyzer. Things being what they are all that one does on the Web include bundles. For instance,

each Site page that we get comes as a progression of parcels, and each email we send leaves as a progression of bundles. In this paper we will for the most part learn about nature of header and payload going in the parcels. Every parcel conveys the data that will help it get to its goal i.e. the sender's IP address, the planned beneficiary's IP address, something that tells the system what number of bundles this message has been broken into and the quantity of this specific parcel. Every bundle contains some portion of the body of the message. Most parcels are part into two sections:

A. ***Header****:* The header contains directions about the information conveyed by the parcel. For instance Length of the Packet, parcel number, convention, goal address, source address and so forth. Substance of header relies on upon transport convention, i.e. [2] TCP,UDP or ARP.

B. ***Payload:*** Payload is likewise called the body or information of a packet. This is the real information that the bundle is conveying to the goal. On the off chance that a packet is settled length, then the payload might be cushioned with clear data to make it the correct size. Data going in payload is scrambled or plain relies on upon the sort of convention utilized for information transmission like telnet or SSH, and so forth.

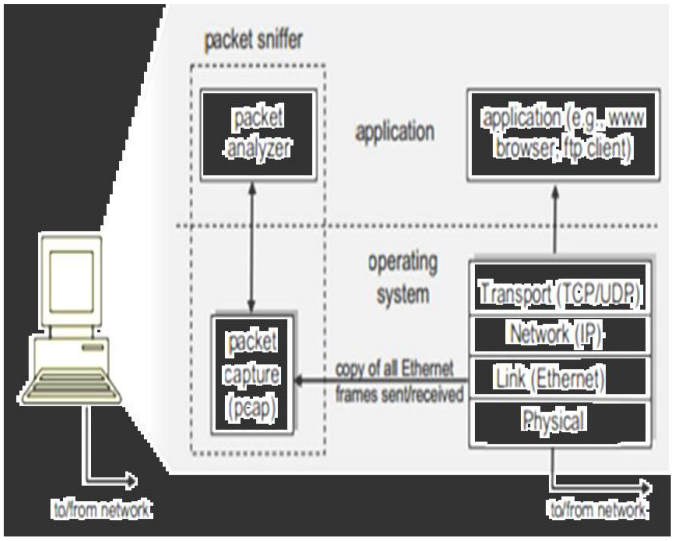

Fig. 1. structure of packet sniffer

The structure of packet sniffer consists of two parts:- packet analyser and packet capture(pcap).Packet analyser works on application layer whereas pcap captures packet from all other layers such as physical layer, link layer, IP

and transport layer.Packet analyser communicates with the pcap which further captures packets from the applications running on the network. Figure 1 shows the basic structure of packet sniffer [3].

Fig. 2. data encapsulation in a packet

Most of the packet sniffers [6] work as a pcap application. The normal flow in a pcap application is to initialize network interface, then further set the filter, to filter the packets to be accepted and rejected. Packets are accepted and log is maintained continuously until the interface is closed, and further processes the packets captured.

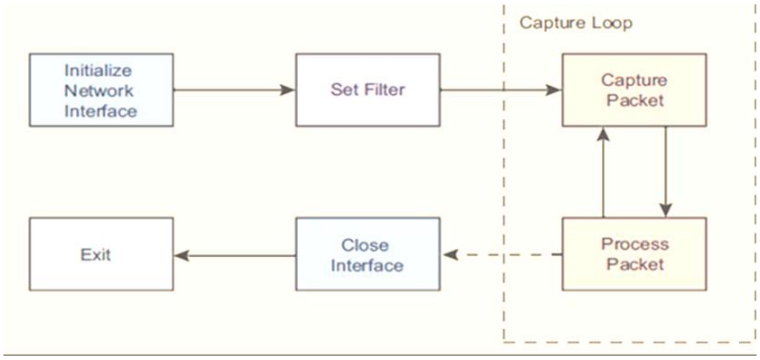

Fig. 3. Normal program flow of a pcap application

To capture the information in these packets it does the following steps [4]:-

Step 1: Initially a socket is created. To deal withdraw binary data, raw sockets are created. For each Socket created it have a socket handle, socket type, local and remote address.

Step 2: Then the NIC (network interface card) is set to a promiscuous mode. Dictionary meaning of promiscuous mode is demonstrating an unselective approach. All packets moving in a network reaches the NIC of all the nodes and then further checks IP address of the destination node and IP address of the current node. Hence, when promiscuous mode is active it accepts all the packets arriving on its NIC irrespective of the destination address.

Step 3: Final step is protocol interpretation. Protocol interpretation means the data to be fetched for the protocols mentioned such as TCP, IP, UDP,ICMP, etc.

This paper gives a strategy to concentrate parcel header and payload data and utilize it to recreate the correspondence. On the off chance that the information is moved in plain content arrangement, then the entire content can be recovered.

To catch parcels going in a system, one can utilize TCPDUMP for UNIX and WINDUMP for windows. TCPDUMP can give extremely itemized data about any system discussion between two machines. Favorable position of TCPDUMP[1] is that it can be run remotely through a SSH or TELNET session, and the machine running TCPDUMP does not need to run x-windows. The application utilizes next to no overhead, since it's a non-graphical interface.

Running TCPDUMP with no of its choice will give data just about header including source and goal Macintosh address, IP address, port number, length, and so on. To get point by point data of parcel including payload or client information, alternative '- x' is utilized with TCP dump.

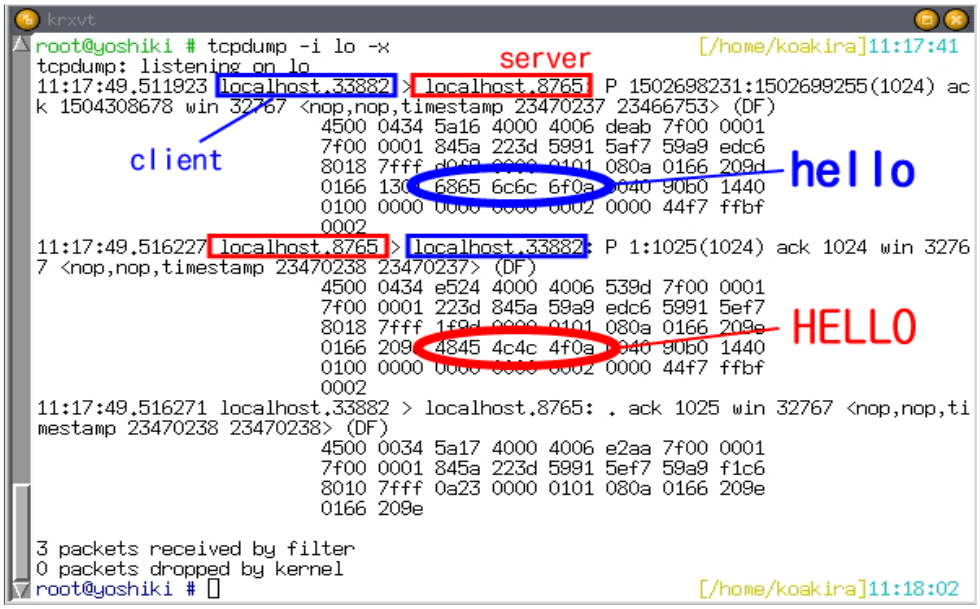

Fig.3 A captured packet with option –x will look

A captured packet with option –x will look like:

15:52:42.475432 00:1d: 09:46:a3:43 > 00:08:02:ee: 1c: 08, ether type IPv4 (0x0800), length 74: 172.31.9.56.41120 > 172.31.9.84.*23* S 2754605757:2754605757(0) win 5840 <mss 1460,sackOK,timestamp 99293 0,nop,wscale 7>

0x0000: 0008 02ee 1c08 001d 0946 a343 0800 4510

0x0010: 003c 7160 4000 4006 5e81 ac1f 0938

0x0020: 0954 a0a0 0017 a42f f2bd 0000 0000

0x0030: 16d0 0ae2 0000 0204 05b4 0402 080a 0001

0x0040: 83dd 0000 0000 0103 0307

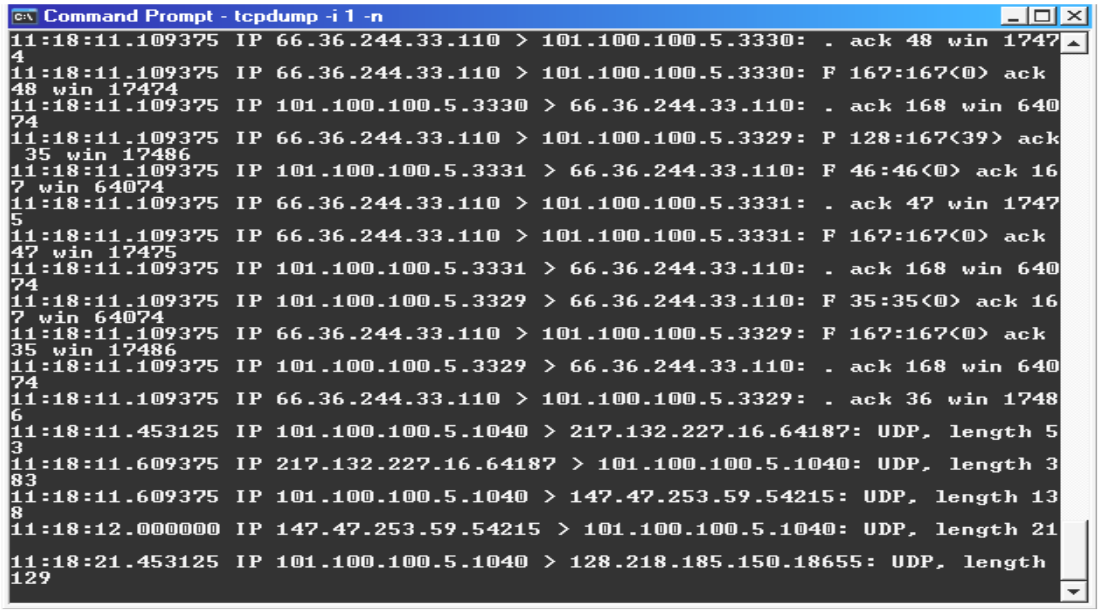

Fig.43 A captured packet in command prompt –TCP dump – i 1 -n

Till 0x300, packet contains header data. From line 0x400 it contains payload data. Payload data of parcel information can be moved in various conventions in scrambled or plain frame.

The packet catch should be possible on a mirror port of a Switch or a Switch through which the movement to be checked is passing.

II. PLAIN CONTENT EXCHANGE APPLICATIONS

The accompanying subsections detail how correspondence between two hubs can be remade for plain content applications like TELNET, FTP, HTTP and SMTP [2]. Just significant data of the packet header/payload has been appeared (set apart with dark foundation) alongside its correspondence elucidation

A. Telnet:

Telnet is a system convention utilized on the Web or neighborhood to give a bidirectional intelligent interchanges office. Commonly, telnet gives access to a charge line interface on a remote host by means of a virtual terminal association which comprises of a 8-bit byte arranged solid information association over the Transmission Control Convention (TCP) and utilizations port number 23. Client information is scattered in-band with TELNET control data. Appeared underneath is a run of the mill Telnet correspondence:

17:03:54.612448 00:1d:09:46:a3:43 > 00:08:02:ee:1c:08, ethertype IPv4 (0x0800), length 74: **172.31.9.35**.42475 > **172.31.9.184**.**23**: S 1458670524:1458670524(0) win 5840 <mss 1460,sackOK,timestamp 4084866 0,nop,wscale 6>

Host with IP 172.31.9.35 telnets to host 172.31.9.184 on port 23.

0x0040: 54e0 6c6f 6769 6e3a 20	T.**login**:.
0x0040: 1f6f 75	.o**u**
0x0040: 57f3 75	W.**u**
0x0040: 2280 73	".**s**
0x0040: 5871 73	Xq**s**
0x0040: 22fe 65	".**e**
0x0040: 5902 65	Y.**e**
0x0040: 238f 72	#.**r**
0x0040: 599b 72	Y.**r**
0x0040: 2428 31	$(**1**
0x0040: 5a31 31	Z1**1**

Study payload carefully and search for “login” in captured packets. Once found “login” every character which repeats itself after 2 characters is included in username e.g. user1 in this case.

0x0040: 5af9 5061 7373 776f 7264 2066 6f72 2075 Z.**Password.for.u**

0x0040: 2587 75	%.**u**
0x0040: 287e 73	(~**s**
0x0040: 28f1 65	(.**e**
0x0040: 2987 72	).**r**
0x0040: 2a34 31	*4**1**

For password look for “Password for.<<initial of username(like ‘u’ in this case)>>”. Then every fourth character is included in password. Here the password is user1.

0x0040: 614d 6c6f 6769 6e3a 2043 616e 6e6f 7420 aM**login:.Cannot.**
0x0040: 6153 4c61 7374 206c 6f67 696e 3a20 5765 aS**Last.login:.We**
0x0040: 6154 1b5d 303b 7573 6572 3140 6c6f 6361 aT.]0;**user1@loca**
0x0040: 6186 5b75 7365 7231 406c 6f63 616c 686f a.[**user1@localho**

If you are logged in successfully, it will show “Last.login:” after “login:. Cannot.” and then “<<username>>@loca” else if you have entered a wrong password it, will show “wrong password” and then again every fourth character will show new entered password.

0x0040: 2c15 63 ,.**c**
0x0040: 64b2 63 d.**c**
0x0040: 2f40 64 /@**d**
0x0040: 653b 64 e;**d**
0x0040: 3079 75 0y**u**
0x0040: 67d8 75 g.**u**
0x0040: 3266 73 2f**s**
0x0040: 6872 73 hr**s**

An entered command is also read as username i.e. every character that repeats itself after two chars. Like command entered here is “cd us”

0x0040: 6ca0 1b5d 303b 7573 6572 3140 6c6f 6361 l..]0;**user1@loca**
0x0040: 372f 6c 7/**l**
0x0040: 704e 6c pN**l**
0x0040: 3add 73 :.**s**
0x0040: 7198 73 q.**s**

Another command entered here is “ls” and following captured packet shows presence of user file in this directory.

0x0040: 7242 1b5b 3030 6d1b 5b30 306d 7573 6572 rB.[00m.[00m**user**

0x0040: 7246 1b5d 303b 7573 6572 3140 6c6f 6361 rF.]0;**user1@loca**

x0x0040: 3cd5 65	<.**e**
0x0040: 75cf 65	u.**e**
0x0040: 405d 78	@]**x**
0x0040: 76d2 78	v.**x**
0x0040: 4161 69	Aa**i**
0x0040: 77d2 69	w.**i**
0x0040: 4261 74	Ba**t**
0x0040: 7898 74	x.**t**
0x0040: 7992 6c6f 676f 7574 0d0a	y.**logout**..

User now enters an "exit" command and a "logout" prompt shows command executed successfully.

B. HTTP:

HTTP (Hyper Text Transfer Protocol) is the Internet application-layer convention. HTTP is executed in two projects: a customer program and server program. The customer program and server programs, executing on various end frameworks, converse with each other by trading HTTP messages. HTTP characterizes the structure of these messages and how the customer and server trade the messages.

There are two sorts of HTTP messages, ask for messages and reaction messages, both of which are talked about underneath.

HTTP Request Message: Below we provide a typical HTTP request message:

GET /somedir/page.html HTTP/1.1
Connection: close
User-agent: Mozilla/4.0
Accept: text/html, image/gif, and image/jpeg
Accept-language:fr
(extra carriage return, line feed)

HTTP Response Message: Below we provide a typical HTTP response message:

HTTP/1.1 200 OK
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 09:23:24 GMT
Content-Length: 6821
Content-Type: text/html

data data data data data ...

Now when an end user request for a page from server, he/she gets a reply in which data is transferred using Unicode [4] encoding. To get ascii translated data one should use **TCPDUMP with option –A** (capital a).

To view all http traffic one is required to capture data tcp port **80** (port used by http). A sample captured packet sequence when a page ad.html is downloaded from a server is:

18:27:28.290646 00:1d:09:46:a3:43 > 00:08:02:ee:1c:08, ethertype IPv4 (0x0800), length 74: **172.31.9.35**.52996 > **172.31.9.184.http**: S 2305058403:2305058403(0) win 5840 <mss 1460,sackOK,timestamp 1101663 0,nop,wscale 6>

...`.1..**GET /ad.html HTTP/1.1**

Host: 172.31.9.184

The above captured packets show that http request is sent from 172.31.9.35 to 172.31.9.184 to “get” ad.html file. “Host” represent host of requested page

18:27:28.292160 00:08:02:ee:1c:08 > 00:1d:09:46:a3:43, ethertype IPv4 (0x0800), length 250: **172.31.9.184.http > 172.31.9.35.52996**: P 1:185(184) ack 480 win 1716 <nop,nop,timestamp 506569665 1101664>

.<html>

<body>
this is 2nd test

Date: Wed, 19 Aug 2009 07:00:02 GMT

Above captured packets show full html coding of requested page and date of request.

..GET /ad.html HTTP/1.1

HTTP/1.1 304 Not Modified

When same page is requested again without modifying it in host machine it shows "NOT MODIFIED" prompt as page is stored in cache and is not modified in main machine.

Hence it can be clearly seen that we can extract complete http communication (web page text) successfully.

C. FTP :

FTP (File Transfer Protocol) [3] allows a user to transfer files to and from a remote network server. FTP uses TCP port number **21.**

Following are some captured packets and their interpretation for ftp communication:

18:08:47.524931 00:1d:09:46:a3:43 > 00:1e:8c:c7:43:7f, ethertype IPv4 (0x0800), length 66: **172.31.9.53**.37901 > **172.31.200.202**.**21**: . ack 1 win 46 <nop,nop,timestamp 5439222 839696>

0x0040: fef6 0000 0000 0103 0307 .....

18:08:47.524872 00:1e:8c:c7:43:7f > 00:1d:09:46:a3:43, ethertype IPv4 (0x0800), length 74: **172.31.200.202**.**21** > **172.31.9.53**.37901: S 3906564666:3906564666(0) ack 857987725 win 5792 <mss 1460,sackOK,timestamp 839696 5439222,nop,wscale 6>

0x0040: d010 0052 fef6 0103 0306 ...R.....

Above packets show communication between 172.31.9.53 (client) and 172.31.100.102 (server) to show connection established between ftp client and server.

18:08:47.526526 00:1e:8c:c7:43:7f > 00:1d:09:46:a3:43, ethertype IPv4 (0x0800), length 86: 172.31.200.202.21 > 172.31.9.53.37901: P 1:21(20) ack 1 win 91 <nop,nop,timestamp 839697 5439222>

0x0040: fef6 3232 3020 2876 7346 5450 6420 322e ..**220.(vsFTPd.2**.

18:08:47.526704 00:1d:09:46:a3:43 > 00:1e:8c:c7:43:7f, ethertype IPv4 (0x0800), length 72: 172.31.9.53.37901 > 172.31.200.202.21: P 1:7(6) ack 21 win 46 <nop,nop,timestamp 5439224 839697>

0x0040: d011 5359 5354 0d0a ..SYST..

Above packets show a positive response by "220.(vsFTPD.2." Here VSFTPD is the FTP server for UNIX or LINUX and 172.31.9.53 replied with a "SYST".

0x0040: fef8 3533 3020 506c 6561 7365 206c 6f67 ..**530.Please.log**
0x0040: d012 5553 4552 2061 6e6f 6e79 6d6f 7573 ..**USER.anonymous**

Promt for username and username can be successfully captured from "USER.<<username>>" (here anonymous login has been used).

0x0040: 1297 3333 3120 506c 6561 7365 2073 7065 ..**331.Please.spe**
0x0040: e3b0 5041 5353 206c 6970 690d 0a ..**PASS.preeti**

Then it prompts to specify password with an 331 request and password can be extracted from "PASS.<<password>>."

0x0040: 1eb4 3233 3020 4c6f 6769 6e20 7375 6363 ..**230.Login.succ**

If logged in successfully it will show "LOGIN.succ"

0x0040: c80f 5459 5045 2041 0d0a ..**TYPE.A**..
0x0040: 193a 3230 3020 5377 6974 6368 696e 6720 .:**200.Switching**.

Above packets show command to set type to ASCII. ASCII is set to send text and binary to send images or executable files. A "200.Switching" prompt conforms switching to ASCII.

0x0040: ea63 5155 4954 0d0a .c**QUIT**..

0x0040: 81dc 3232 3120 476f 6f64 6279 652e 0d0a ..**221.Goodbye**...

A “Quit” command is followed by a confirmation of “221.Goodbye” to show logged out successful.

Using the above, it is possible to reconstruct the complete ftp session.

D. SMTP:

SMTP (Simple Mail Transfer Protocol) is the application layer protocol for exchanging email. It uses TCP port 25. A sample captured packet sequence when mail messages are exchanged is:

15:52:26.250270 00:1d:09:46:a3:43 > 00:18:71:e5:47:82, ethertype IPv4 (0x0800), length 753: 172.31.9.35.52112 > 172.31.1.227.**squid**: P 1:688(687) ack 1 win 92 <nop,nop,timestamp 1746526 299668948>

GET http://mail.google.com/mail/ HTTP/1.1

Host: mail.google.com

HTTP/1.0 302 Moved Temporarily

Set-Cookie: GMAIL_RTT=EXPIRED; Expires=Thu, 20-Aug-2009 04:52:43 GMT; Path=/mail

.....**DCONNECT www.google.com:443 HTTP/1.1**

User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.1b4) Gecko/20090427 Fedora/3.5-0.20.beta4.fc11 Firefox/3.5b4

....HTTP/1.0 200 Connection established

CONNECT mail.google.com:443 HTTP/1.1

User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.1b4) Gecko/20090427 Fedora/3.5-0.20.beta4.fc11 Firefox/3.5b4

The above packets show connection or disconnection prompt when tried to access to www.gmail.com User-agent represents browser used to access gmail webclient. The requests to transfer mail passes through squid proxy server.

.....GET

http://www.google.co.in/accounts/SetSID?ssdc=1&sidt=iJROOyMBAAA%3D.g%2BiyA9aas6v72XiQLN2hWw%2FKFZsIdJHcFPZ2Ft7ylq6yVVFr0G4JQ4yLCH9jsMVU4JQHD%2FYD3zBxHf5aOKkzgFL33VKnmATA7KMUF%2FOfiqO8MPYiBl4OJM0fezW%2BA2GDnbcUP6hHtFvhWwKMjDu1926trI3LKHU8OUtGfmnnLORgt8ZRu9

kMtNa5LnVSBazTe3%2BG3f47deceK2ZCmqClBkYdMbo7yjS8U%2FnEL7OvsNuuBgTVWMGaetK2AAJ9zLdV
Z6z5esNnND4UapBJ%2Fq0lQ6dqqr2D2ZKIZ%2B1LyirX7eTk3JNI6qhnqjpdn8BexIZdjcR3lRf6rlIFMeXTQI%2B
d0HHVkiOynRRU39yu3eCsFUo%3D.sAIZz1T8pMcxXwliRttJWg%3D%3D&continue=http%3A%2F%2Fmail.go
ogle.com%2Fmail%2F%3Fauth%3DDQAAAK8AAADhuWAX62_ZchB9vMj6E3jFeIhQY7wZ3X9VhyvAXLHP3
RfI2paRFpKoeYeDlLo5m5fxCUZKFr2A_v-u-
LUV13JU2pXUhAez2ovG2MKsoUZfjf3bskVBoEx6_YkXowKRF5H6hiqT1tD9MwnwdnYzoLof9k9tzGi-
PypgDKPUQojnRaPU8oju_NijS5T-C6kAEUwtQnEgJ-
31B4oM2di0qy1ZFftOeMJojJwut83UFStkNw%26gausr%3D**lipi.mishra.200103%2540gmail.com** HTTP/1.1

Host: www.google.co.in

The above packets show that the inbox of user lipi.mishra.200103 has been obtained.

GET http://mail.google.com/mail/?shva=1 HTTP/1.1

Shows user gets into inbox.

Host: mail.google.com

.....HTTP/1.0 200 OK

Cache-Control: no-cache, no-store, max-age=0, must-revalidate

.....HTTP/1.0 200 OK

Cache-Control: public

.....**POST**
http://mail.google.com/mail/?ui=2&ik=0d7d4c8f3e&view=cv&th=12336d14aa0a824a&th=123338a12f9420dd
&th=12332b5ab8b122e4&th=1232c13f0e87e8f1&th=12324bf3e39c7711&prf=1&nsc=1&rt=j&search=inbox
HTTP/1.1

Host: mail.google.com

Proxy-Authorization: Basic dmtnOm1pbmkxMjM=

HTTP/1.0 204 No Content

Content-Type: text/html; charset=UTF-8

E...h.@.@.h..... #.8..@..:aWR....[.;.....

.....B:.Od..6...y/..#...@a...8.....*(..U.K...%.a.dVyK.._.J..^...qJE..u|KH.O. ..RM.....h!4s.?.....+ [..S...E0c..gN7e....,..c_..t....&U..&?mX7.N].....W..([...H.....y.28.....0.x..0(...0....kX.....2b~.....(L.V..0Bj..Wl.%lZ.RM }.....R...-+..O..7...x...].L.v.h>uD]...00...yc.h.Rbr.2.2P.).6.A>!...*y..... .N.....$..|.....)..|...*...}...2H&^.i.?..'...w'.>.v.K. mGI.v...c.i.....v.....^.....:...3..:....&....6...j.<7../.A...v....k.x.^...8~..`...78..

UTF-8 encypted inbox.

.**GET**

http://pagead2.googlesyndication.com/pagead/adview?ai=BMbqYIiiOSoKtBI7GqgOAovSBBN2_qXGH7MaTCMCNtwGwmewPEAEYASCGj4ACOABQwLOtvP_____AWDlgumD4A6yAQ9tYWlsLmdvb2dsZS5jb226AQhnbWFpbC10bMgBAdoBNmh0dHA6Ly9tYWlsLmdvb2dsZS5jb20vZ243M2o0aXR6MmtxcncxdzB1bGpjZHc1YTNkNnlmN4ACAakC4TkPHLgLVj7IAoeElwioAwHoA-AF6AMF9QMAAAAE&sigh=2L0vM59IPAY HTTP/1.1

Host: pagead2.googlesyndication.com

Host: chatenabled.mail.google.com

Last-Modified: Mon, 10 Aug 2009 18:44:46 GMT

[[26,["r",1,[[0,"","2","",20,0,0,1,"dnd"," Bleeding packet - Leona Lewis",0,2,2,1,0,"**madan.aanchal.11**","madan.aanchal.11",0,"",1,0,0,"",0,0,," Bleeding packet- Leona Lewis","5d9266dc0e330230",[]

The above packets show a Gtalk message in the inbox from “madan.aancha.11” and the message as ”Bleeding Love - Leona Lewis”.

[[30,["nm","1233b4d0f39edc20","1233b4d0f39edc20","**sent","preeti raj**","0",["^f","^all"]

[[31,["r",1,[[0,"","0","",228,0,0,0,,"",0,2,2,0,0,"","**amber**",0,"",0,0,0,"",0,0,,"","19",[]

[[33,["qi",0,40,27]

.....HTTP/1.0 200 OK

Cache-Control: no-cache, no-store, max-age=0, must-revalidate

...$...b.mb....6.\.n..%o...`..L]{.....1.9...&.CQ{.'v.].S..J.B..?.&+....+...V6E.Q.n....-
...a!...z...0..7....k..C.v....=.....|.7.h...{.....#x..O.V..6:y.....~.."..EP../.F.6.iX8Q..\.@.z..~..Ej...@.Y.....a...C.}..E.....
.)...>.=../w.d...

The above packets show UTF-8 encoded mail sent to “amber” from lipi.mishra.200103@gmail.com. Again tcpdump –A option can be used to obtain the message in plain text format.

Hence it can be clearly seen that we can extract complete smtp communication successfully if the mail messages are not encrypted.

III. ENCRYPTED CONTENT EXCHANGE

The standard framework applications like telnet, ftp or http are profitable however characteristically unsafe, since they all make you send a mystery key and data in clear substance over an inflexibly dangerous framework. **Ssh**[10[, **sftp** and https are secure adjustments of these traditions. Here we will analyze https and ssh applications.

The customary system applications like telnet, ftp or http are advantageous yet inalienably risky, since they all make you send a secret word and information in clear content over an inexorably hazardous system. **ssh, sftp** and https are secure variants of these conventions. Here we will examine https and ssh applications.

A. HTTPS:

HTTPS is secure version of HTTP and uses tcp port 443.

Shown below are some packets captured during https connection (the same page is requested as in http earlier):

18:24:10.445345 00:1d:09:46:a3:43 > 00:08:02:ee:1c:08, ethertype IPv4 (0x0800), length 74: **172.31.9.35**.49962 > **172.31.9.184.https**: S 3509873094:3509873094(0) win 5840 <mss 1460,sackOK,timestamp 903817 0,nop,wscale

6>

18:24:25.675549 00:08:02:ee:1c:08 > 00:1d:09:46:a3:43, ethertype IPv4 (0x0800), length 103: **172.31.9.184**.**https** > **172.31.9.35**.49962: P 3187:3224(37) ack 1552 win 2520 <nop,nop,timestamp 506387016 904050>

E..Y.>@.@.=G.. ... #...*.Q.T.4o... .....

...H.
.r.... ..;'.y.;...+.....G+?.A...#.$\8k.

18:24:28.627167 00:1d:09:46:a3:43 > 00:08:02:ee:1c:08, ethertype IPv4 (0x0800), length 103: **172.31.9.35**.49962 > **172.31.9.184**.**https**: P 1552:1589(37) ack 3225 win 408 <nop,nop,timestamp 922000 506387016>

E..Y.\@.@.,).. #.. ..*...4o..Q.z....ke.....

.....H.... g...R.=:D@..G.%NIgp.WJ....x.1>5.

These packets show that https request is sent from 172.31.9.35 to 172.31.9.184. Captured packet payloads show text transferred in encrypted form.

It can be seen that we can extract only the https connection details and it is not possible to extract the complete communication as the traffic flowing is encrypted.

B. SSH

SSH (Secure Shell) gives remote shell get to safely [5]. By utilizing SSH, the activity is scrambled and one can make 'man-in-the-center' assaults practically unimaginable. It likewise shields from DNS and IP mocking. As a reward, it offers the likelihood to pack the movement and accordingly make exchanges quicker. Seen from the customer level, SSH gives two levels of validation. SSH utilizes open key cryptography to confirm the remote PC and enables the remote PC to validate the client, if vital. It utilizes TCP port 22.

When a user first time connects to an IP using ssh, it shows something like:

ssh 172.31.9.55

The authenticity of host '172.31.9.55 (172.31.9.55)' can't be established.

RSA key fingerprint is **45:06:dd:de:d8:95:65:27:80:2e:7b:26:bc:b7:e1:1e**.

Are you sure you want to continue connecting (yes/no)?

yes

Warning: Permanently added '172.31.9.55' (RSA)[9] **to the list of known hosts.**

root@172.31.9.55's password:
Last login: Thu Jun 18 20:16:48 2009
It adds IP with RSA [6] public key in location **~/.ssh/known_hosts**. Hence when same user tries to ssh on same IP he/she does not require to go through these authentication.

Public key is saved in format as:
AAAAB3Nzac1yc2EAAAABIwAAAQEA<<then the key>>==

In public-key cryptography, a **public key fingerprint**[8] is a short sequence of bytes used to authenticate or look up a longer public key. Fingerprints are created by applying a cryptographic hash function to a public key. When displayed for human inspection, fingerprints are usually encoded into hexadecimal strings.
For example, a 128-bit MD5 fingerprint for SSH would be displayed as follows:
45:06:dd:de:d8:95:65:27:80:2e:7b:26:bc:b7:e1:1e
If we captured SSH packets, it will be like:

16:03:06.212706 00:1d:09:46:a3:43 > 00:21:9b:f4:31:02, ethertype IPv4 (0x0800), length 74: **172.31.9.158.47408** > **172.31.9.88.*22***: S 3948140751:3948140751(0) win 5840 <mss 1460,sackOK,timestamp 769924 0,nop,wscale 7>

0x0000: 0021 9bf4 3102 001d 0946 a343 0800 4500 .!..1....F.C..E.
0x0010: 003c 674b 4000 4006 683c ac1f 099e ac1f .<gK@.@.h<.....
0x0020: 0958 b930 0016 eb53 d8cf 0000 0000 a002 .X.0...S.....

0x0030: 16d0 8901 0000 0204 05b4 0402 080a 000b .....

0x0040: bf84 0000 0000 0103 0307

0x0040: bf86 5353 482d 322e 302d 4f70 656e 5353 ..SSH-2.0-**OpenSS**

From the captured packets, it can be concluded that the ssh connection was established from 172.31.9.158 to 172.31.9.88 and SSH2.0 has been used.

It can be seen that we can extract only the ssh connection details and it is not possible to extract the complete communication as the traffic flowing is encrypted.

IV. RESULT

In this analysis to develop an IPv4/IPv6 protocol analyzer[7] or packet sniffer. It is a web based computer application that can intercept and log traffic passing over a digital network or part of a network. As data streams flow across the network, the sniffer captures packets using TCPDUMP and, if needed, decodes the packet's raw data, showing the values of various fields in the packet, and analyzes its content according to the appropriate logical operator or other specifications. When traffic is captured, either the entire contents of packets can be recorded, or the headers can be recorded without recording the total content of the packet.

V. CONCLUSION

As we have appeared in this paper, it is conceivable to catch parcels and utilizing the data in the header and payload of the bundles, remake the correspondence between any two hubs in a system. The parcel payload either goes in encoded frame or plain content shape. It has been additionally indicated how one can get the content of the data going in payload if convention of correspondence uses plain content organization. The total correspondence remaking philosophy for telnet, ftp, http and smtp has been illustrated. For https and ssh, it has been demonstrated that lone data about the imparting hubs can be known.

ACKNOWLEDGEMENT

The authors wish to thanks, Er. Navpreet Singh(IIT Kanpur), Er. Ram Niwas Sharma (AKTU) and Dr. Beenu Raj (CSIR Delhi) for providing valuable comments the research presented in this paper.

REFERENCES

[1] ANSHUL GUPTA, SURESH GYAN VIHAR "A Research Study on Packet Sniffing Tool TCPDUMP " International Journal of Communication and Computer Technologies Volume 01 – No.49 Issue: 06 Jul 2013 ISSN NUMBER : 2278-9723.

[2] Mahesh Kumar, Rakhi Yadav, "TCP & UDP PACKETS ANALYSIS USING WIRESHARK", International Journal of Science, Engineering and Technology Research (IJSETR), Volume 4, Issue 7, July 2015

[3] L. Garcia,"programming with libpcap," in Hacking- Practical protection hard core IT magazine, Vol 3, 2008, pp. 38-46.

[4] "Tutorial on Wireshark". Internet: http://webhost.bridgew.edu/sattar/CS430/HW/ LABS/wireshark.htm [Oct. 10,2013].

[5] [Tools List] Les Cottrell "Network Monitoring Tools" http://www.slac.stanford.edu/~cottrell/tcom/nmtf_tools.html A good list of network monitoring tools.

[6] P. Wu, Y. Cui, J. Wu, J. Liu, C. Metz, "Transition from IPv4 to IPv6: A State-of-the-Art Survey", IEEE Communications Surveys & Tutorials, Vol. 15, no. 3, pp 1407 – 1424, 2012.

[7] https://343networks.files.wordpress.com/2010/06/ipv4-ipv6-header.gif

[8] M.ManiRoja† and Sudhir Sawarkar††, "Biometric Database Protection using Public Key Cryptography", IJCSNS International Journal of Computer Science and Network Security, VOL.13 No.5, May 2013.

[9] Mofeed Turky Rashid1 , Huda Ameer Zaki2, "RSA Cryptographic Key Generation Using Fingerprint Minutiae", Iraqi Commission for Computers & Informatics (ICCI) Iraqi Journal for Computers and Informatics (IJCI) Vol (1) Issue (1), 2014.

[10] http://www.net4society.eu/public/reports.php.