IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

PACKET ANALYSIS USING PACKET SNIFFERS

Jyoti

Senior Engineer, Bharat Electronics Limited (India)

ABSTRACT

In the today technology advancing world everything is getting automated to be integrated centrally. The size of the integrated systems is increasing day by day at a very fast pace. So there is need to tools to help this integration fast and error free. To meet this demand there are many network sniffer tools that will help in integration of these vast systems.

Keywords: Packet Capture; Packet Sniffer; Network Monitoring; Wireshark; NIC

I. INTRODUCTION

A packet sniffer is a program that can see all of the information passing over the network it is connected to. As data steams back and forth on the network the program looks at it or 'sniffs' each packet. A packet is a part of a message that has been broken up. Packet analysis can help us understand network characteristics, learn who is on network, determine who or what is utilizing available bandwidth, identify peak network usage times, identify possible attacks or malicious activity, and find unsecured and bloated applications[1].

II. WORKING

There are two modes in which a network interface of a machine work i.e Promiscuous and Non-promiscuous. Promiscuous mode in one in which the NIC of the machine can take over all packets and a frame it receives on network, namely this machine (involving its software) is sniffer. As shown in the following diagram*[2]*

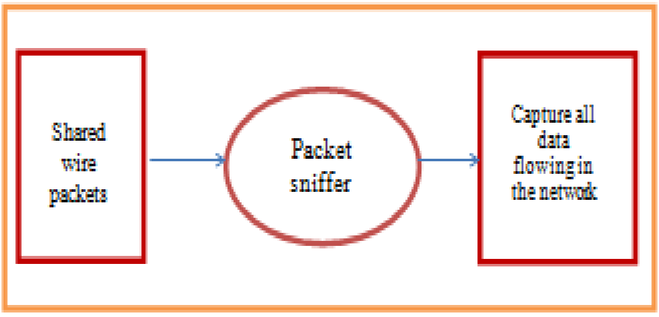


Fig. 1 Promiscuous Mode

When a data packet is transmitted in non-promiscuous mode, all the LAN devices "listen to" the data to determine is the network address included in the data packet is theirs. If it isn't, the data packet is passed onto the next LAN device until the device with correct network address is reached. That device receives and reads the data. As shown in the diagram:-

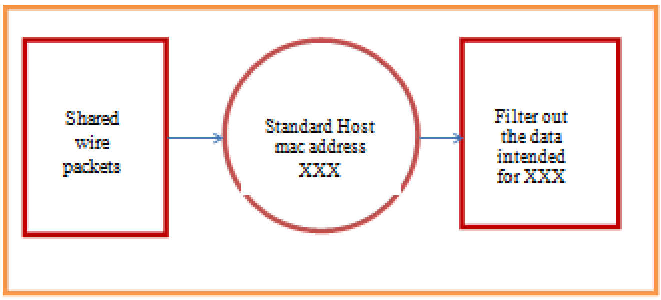

Shared wire packets
Standard Host mac address XXX
Filter out the data intended for XXX

Fig. 2 Non- Promiscuous Mode

2.1 Components

Sniffer is a combination of hardware and software. Different sniffers may have various configurations on account of designation and final usage, but basically, a sniffer is composed of four parts:

- **Hardware: -** most sniffing products can work with standard adapters. Some sniffers only support Ethernet or wireless adapters whereas others support multi-adapters and allow customization.
- **Drive program: -** this is a core component of a sniffer. Each sniffing product has its own drive program, only after completing installation can a sniffer start to capture traffic and data from network.
- **Buffer: -**a buffer is a storage device for captured data from network. In general, there are two modes of buffers: keep capturing until the storage place full, or keep capturing and overflowing as the latest captured data keep replacing the oldest data.
- **Packet Analysis:-**packet analysis can be done on real time or we can analyze packets after storing it. We can analyze both header and actual data, when we store data in memory or we perform real time analysis, decoder is used to decode the data store in packets.

2.2 Working Principle

When a computer sends a data to the network, it sends in the form of packets. These packets are the blocks of data that are actually directed to the certain deputed system. Every sent data has its receiving point. So, all the data are directly handled by specific computer. A system reads and receives only that data which only that data which is intended for it. The packet sniffing process involves a collaborate effort between the software and the hardware. This process is broken down into three steps.

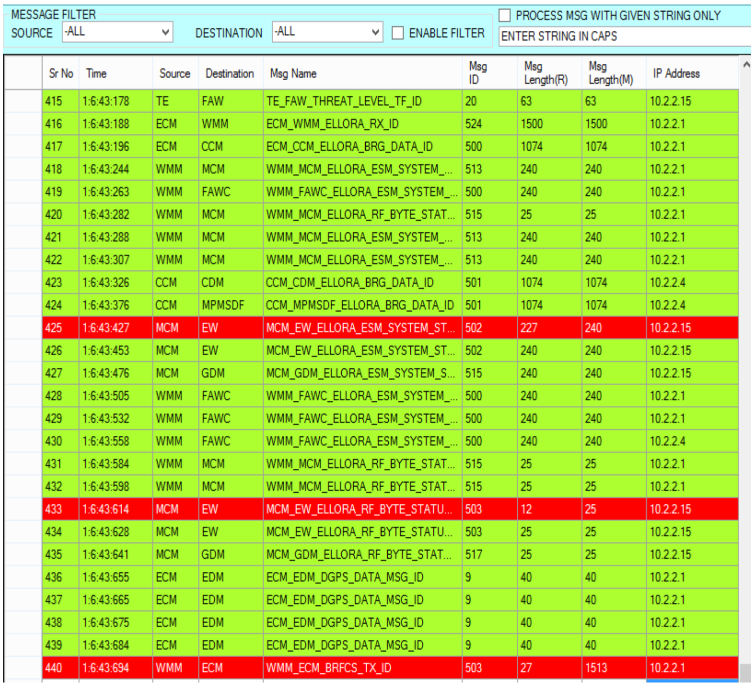
1. Packet sniffer collects raw binary data from the wire. Normally this is done by switching the selected network interface into unrestrained mode.
2. The collected binary data is converted into readable form.
3. The packet sniffer collected all data, verifies its protocol and begins its analysis.[3]

III RELATED WORK

Integrating huge software projects with the help of the current network sniffing tools is a very tedious task. Thousands of messages flow in the integrated system. And around hundreds of software exes of different modules interact with each other to perform one single goal. To perform this huge integration possible in least possible time we need a sniffing tool specifically according to our project. Because the current available tools will only show the byte format of messages then to calculate the byte and find of the error is very tough task especially when there are thousands of messages flowing. Also it is a tedious to specifically isolate the required message among the thousands of other messages flowing in the network.

IV PROPOSED WORK

In this proposed tool, we just need to make the text files containing entries for all messages that will be integrated with the help of this tool. There are entries for all the software exes in the ‘TO’ and ‘FROM’ dropdown box as shown in the figure below. By default no entries are selected and all the messages will be displayed by default. If you want to see the specific ‘TO’ and ‘FROM’ message then that entry should be selected. There is a color coding of the messages on the basis of the length of the message received and the actual length of the message. If there is a mismatch in the length then red color will be shown else it’s a match then the message will be shown in green color. It’s not that this tool will show only the recorded messages but also show the unrecorded messages as unknown event in yellow color.

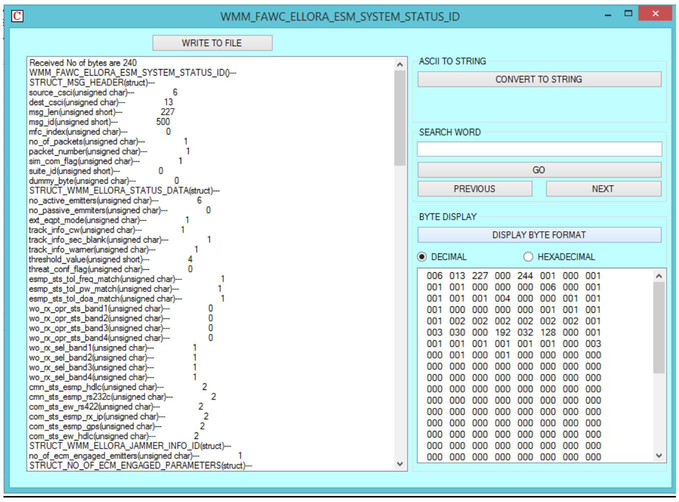


MESSAGE FILTER
SOURCE -ALL   DESTINATION -ALL   □ ENABLE FILTER
□ PROCESS MSG WITH GIVEN STRING ONLY
ENTER STRING IN CAPS

Sr No	Time	Source	Destination	Msg Name	Msg ID	Msg Length(R)	Msg Length(M)	IP Address
415	1:6:43:178	TE	FAW	TE_FAW_THREAT_LEVEL_TF_ID	20	63	63	10.2.2.15
416	1:6:43:188	ECM	WMM	ECM_WMM_ELLORA_RX_ID	524	1500	1500	10.2.2.1
417	1:6:43:196	ECM	CCM	ECM_CCM_ELLORA_BRG_DATA_ID	500	1074	1074	10.2.2.1
418	1:6:43:244	WMM	MCM	WMM_MCM_ELLORA_ESM_SYSTEM_...	513	240	240	10.2.2.1
419	1:6:43:263	WMM	FAWC	WMM_FAWC_ELLORA_ESM_SYSTEM_...	500	240	240	10.2.2.1
420	1:6:43:282	WMM	MCM	WMM_MCM_ELLORA_RF_BYTE_STAT...	515	25	25	10.2.2.1
421	1:6:43:288	WMM	MCM	WMM_MCM_ELLORA_ESM_SYSTEM_...	513	240	240	10.2.2.1
422	1:6:43:307	WMM	MCM	WMM_MCM_ELLORA_ESM_SYSTEM_...	513	240	240	10.2.2.1
423	1:6:43:326	CCM	CDM	CCM_CDM_ELLORA_BRG_DATA_ID	501	1074	1074	10.2.2.4
424	1:6:43:376	CCM	MPMSDF	CCM_MPMSDF_ELLORA_BRG_DATA_ID	501	1074	1074	10.2.2.4
425	1:6:43:427	MCM	EW	MCM_EW_ELLORA_ESM_SYSTEM_ST...	502	227	240	10.2.2.15
426	1:6:43:453	MCM	EW	MCM_EW_ELLORA_ESM_SYSTEM_ST...	502	240	240	10.2.2.15
427	1:6:43:476	MCM	GDM	MCM_GDM_ELLORA_ESM_SYSTEM_S...	515	240	240	10.2.2.15
428	1:6:43:505	WMM	FAWC	WMM_FAWC_ELLORA_ESM_SYSTEM_...	500	240	240	10.2.2.1
429	1:6:43:532	WMM	FAWC	WMM_FAWC_ELLORA_ESM_SYSTEM_...	500	240	240	10.2.2.1
430	1:6:43:558	WMM	FAWC	WMM_FAWC_ELLORA_ESM_SYSTEM_...	500	240	240	10.2.2.4
431	1:6:43:584	WMM	MCM	WMM_MCM_ELLORA_RF_BYTE_STAT...	515	25	25	10.2.2.1
432	1:6:43:598	WMM	MCM	WMM_MCM_ELLORA_RF_BYTE_STAT...	515	25	25	10.2.2.1
433	1:6:43:614	MCM	EW	MCM_EW_ELLORA_RF_BYTE_STATU...	503	12	25	10.2.2.15
434	1:6:43:628	MCM	EW	MCM_EW_ELLORA_RF_BYTE_STATU...	503	25	25	10.2.2.15
435	1:6:43:641	MCM	GDM	MCM_GDM_ELLORA_RF_BYTE_STAT...	517	25	25	10.2.2.15
436	1:6:43:655	ECM	EDM	ECM_EDM_DGPS_DATA_MSG_ID	9	40	40	10.2.2.1
437	1:6:43:665	ECM	EDM	ECM_EDM_DGPS_DATA_MSG_ID	9	40	40	10.2.2.1
438	1:6:43:675	ECM	EDM	ECM_EDM_DGPS_DATA_MSG_ID	9	40	40	10.2.2.1
439	1:6:43:684	ECM	EDM	ECM_EDM_DGPS_DATA_MSG_ID	9	40	40	10.2.2.1
440	1:6:43:694	WMM	ECM	WMM_ECM_BRFCS_TX_ID	503	27	1513	10.2.2.1

Figure: - Proposed Network Tool

Once you click a message a new window will open and all the contents will be displayed with the labels as recorded in the text file displaying the content as forwarded by the software exe as shown in the figure below. This way you can see what actually is being forwarded and what is received because to a message passes through several other software exes before reaching its destination. So this way we can easily find who is manipulating with the data and where actually is the problem and act on it. Had this been done through traditional sniffer then just analyzing a single message passing through few software exes would have taken a lot of time which can be done through this network tool in few minutes.

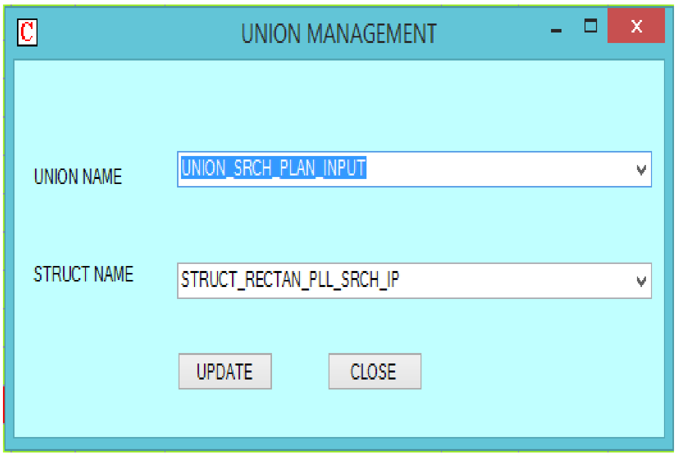


```
Received No of bytes are 240
WMM_FAWC_ELLORA_ESM_SYSTEM_STATUS_ID()---
STRUCT_MSG_HEADER(struct)---
source_csci(unsigned char)---          6
dest_csci(unsigned char)---          13
msg_len(unsigned short)---          227
msg_id(unsigned short)---          500
mfc_index(unsigned char)---          0
no_of_packets(unsigned char)---          1
packet_number(unsigned char)---          1
sim_com_flag(unsigned char)---          1
suite_id(unsigned short)---          0
dummy_byte(unsigned char)---          0
STRUCT_WMM_ELLORA_STATUS_DATA(struct)---
no_active_emitters(unsigned char)---          6
no_passive_emmiters(unsigned char)---          0
ext_eqpt_mode(unsigned char)---          1
track_info_cw(unsigned char)---          1
track_info_sec_blank(unsigned char)---          1
track_info_warner(unsigned char)---          1
threshold_value(unsigned short)---          4
threat_conf_flag(unsigned char)---          0
esmp_sts_tol_freq_match(unsigned char)---          1
esmp_sts_tol_pw_match(unsigned char)---          1
esmp_sts_tol_doa_match(unsigned char)---          1
wo_rx_opr_sts_band1(unsigned char)---          0
wo_rx_opr_sts_band2(unsigned char)---          0
wo_rx_opr_sts_band3(unsigned char)---          0
wo_rx_opr_sts_band4(unsigned char)---          0
wo_rx_sel_band1(unsigned char)---          1
wo_rx_sel_band2(unsigned char)---          1
wo_rx_sel_band3(unsigned char)---          1
wo_rx_sel_band4(unsigned char)---          1
cmn_sts_esmp_hdlc(unsigned char)---          2
cmn_sts_esmp_rs232c(unsigned char)---          2
com_sts_ew_rs422(unsigned char)---          2
com_sts_esmp_rx_ip(unsigned char)---          2
com_sts_esmp_gps(unsigned char)---          2
com_sts_ew_hdlc(unsigned char)---          2
STRUCT_WMM_ELLORA_JAMMER_INFO_ID(struct)---
no_of_ecm_engaged_emitters(unsigned char)---          1
STRUCT_NO_OF_ECM_ENGAGED_PARAMETERS(struct)---
```

```
006  013  227  000  244  001  000  001
001  001  000  000  000  006  000  001
001  001  001  004  000  000  001  001
001  000  000  000  000  001  001  001
001  002  002  002  002  002  002  001
003  030  000  192  032  128  000  001
001  001  001  001  001  001  000  003
000  001  000  001  000  000  000  000
000  000  000  000  000  000  000  000
000  000  000  000  000  000  000  000
000  000  000  000  000  000  000  000
000  000  000  000  000  000  000  000
000  000  000  000  000  000  000  000
000  000  000  000  000  000  000  000
000  000  000  000  000  000  000  000
000  000  000  000  000  000  000  000
000  000  000  000  000  000  000  000
```

Figure: - Message window in the tool

Apart from the main functionality described above there are many other add-ons to this tool. If you want to see the byte format / hexadecimal format in which the message is flowing you can see that also along with the message window displayed in the right. In the message window there is search option also in case the message is too big to search for a particular keyword. Some messages also contain union so there is an option to manage union to see what structure in that union you want to see as shown in the diagram below:-



UNION NAME: UNION_SRCH_PLAN_INPUT

STRUCT NAME: STRUCT_RECTAN_PLL_SRCH_IP

Figure: - Manage union dialog in tool

You take log of the whole set of messages currently received and save that in the text file and just study it for later use as shown in the figure below:-

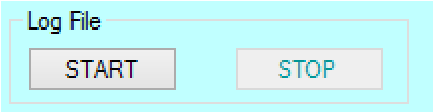
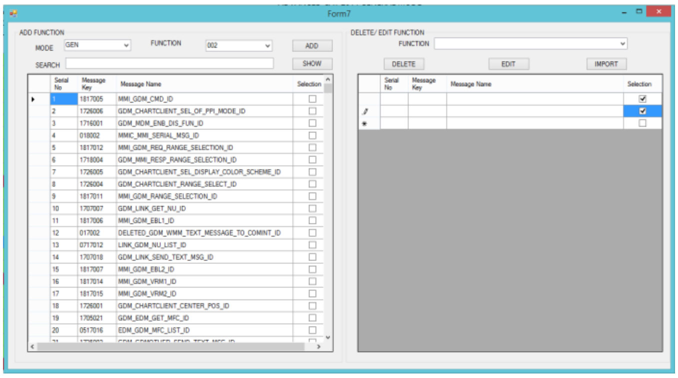


Figure: - Creating Log File in tool

as there are thousands of messages flowing the network, you can specifically select any number of messages you want to enter in a file and only those messages will be displayed, as shown in the figure below:-



MODE GEN

FUNCTION 002

SEARCH

Serial No	Message Key	Message Name	Selection
1	1817005	MMI_GDM_CMD_ID	□
2	1726006	GDM_CHARTCLIENT_SEL_OF_PPI_MODE_ID	□
3	1716001	GDM_MDM_ENB_DIS_FUN_ID	□
4	018002	MMIC_MMI_SERIAL_MSG_ID	□
5	1817012	MMI_GDM_REQ_RANGE_SELECTION_ID	□
6	1718004	GDM_MMI_RESP_RANGE_SELECTION_ID	□
7	1726005	GDM_CHARTCLIENT_SEL_DISPLAY_COLOR_SCHEME_ID	□
8	1726004	GDM_CHARTCLIENT_RANGE_SELECT_ID	□
9	1817011	MMI_GDM_RANGE_SELECTION_ID	□
10	1707007	GDM_LINK_GET_NU_ID	□
11	1817006	MMI_GDM_EBL1_ID	□
12	017002	DELETED_GDM_WMM_TEXT_MESSAGE_TO_COMINT_ID	□
13	0717012	LINK_GDM_NU_LIST_ID	□
14	1707018	GDM_LINK_SEND_TEXT_MSG_ID	□
15	1817007	MMI_GDM_EBL2_ID	□
16	1817014	MMI_GDM_VRM1_ID	□
17	1817015	MMI_GDM_VRM2_ID	□
18	1726001	GDM_CHARTCLIENT_CENTER_POS_ID	□
19	1705021	GDM_EDM_GET_MFC_ID	□
20	0517016	EDM_GDM_MFC_LIST_ID	□

FUNCTION

Serial No	Message Key	Message Name	Selection
			☑
			☑
			□

Figure: - Select message dialog in tool

V RESULT

As is clear from the description above and various features provided with this tool, this tool is way above than the traditional sniffers available for the integration of a specific huge networking project. Suppose there are 1000 of messages that we need to check in a network. Then filtering and analyzing these messages byte wise with traditional sniffers will take on average 15 minutes, depending on the frequency, length etc. of the message. On the other hand in this proposed tool it will hardly take 30 seconds. So will the traditional sniffer it take 15*1000=15000 minutes= 250 hours= 11 days (approx.) that too working day and night. On our proposed tool it will take 30*1000=30,000 seconds= 500 minutes=9 hours (approx.). So this is a huge decrease in time from 11 days to 1/3rd a day. This has saved so much of resources and decrease in the cost of integration. There is no comparison on the basis of precision also as our proposed tool is very precise in comparison to the manual counting and analysis of bytes where messages are as huge as hundreds of byte also. Apart from one time making of text files of the messages there is huge saving of time and precision by using the proposed tool.

VI CONCLUSION

In this paper we have seen that how with the help of this network analysis tool we have made the integration of vast projects like of Indian navy so fast and hassle free. With the help of the tool integration became so transparent which was earlier a very difficult task as to determine which software exe is actually sending the wrong data.

REFERENCES

3 (Chris Sanders and Chris Sanders, “Practical Packet Analysis” Pub. Date: May 17, 2007 ISBN-13:978-1-59327-149-7)

4 Qadeer M.A., Zahid M., Iqbal A., Siddiqui M.R “Network Analysis and Intrusion Detection Using Packet Sniffer ICCSN ' Second International Conference, 2010, Page(s): 313 – 317

5 Pallavi Asrodia* and Hemlata Patel, “Analysis of Various Packet Sniffing Tools for Network Monitoring and Analysis”, International Journal of Electrical, Electronics and Computer Engineering vol.1 no.1 pp. 55-58(2012).

6 S. McCanne and V.Jacobson. "The BSD Packet Filter: New Architecture for User Level Packet Capture", *USENIX Conference*, January, Pages 259-270(1993)..

7 Dulal C. Kar Felix Fuentes. Ethereal vs. tcpdump: A comparative study on packet sniffing tools for educational purpose. Journal of Computing Sciences in Colleges archive, Volume **20**(4), pp 169-176, (2005)..

8 All about capsa [Online] Available www.colasoft.com.

9 All about Tools [Online] Available: http://www. sectools.org.

10 All about soft perfect network protocol analyzer [Online] Available http://www.softperfect.com/products/networksniffer/