ISSN:2231-0711

Aruna Varanasi et al | IJCSET(www.ijcset.net) | December 2016 | Vol 6, Issue 12,406-409

Comparative Study of Packet Sniffing tools for HTTP Network Monitoring and Analyzing

Dr. Aruna Varanasi[#1], P. Swathi[*2]

[1]*Professor and Head, Computer Science and Engineering,*

[2]*Post Graduate Scholars, Computer Science and Engineering.*

Sreenidhi Institute of Science and Technology, Ghatkesar, T.S, India.

***Abstract*— The use of computers has expeditiously increased in the last few decades. Coupled with this has been the exponential growth of the internet. Computers can now exchange the large volumes of information. This has resulted in an ever increasing need for tools and applications that can monitor and analysing the network traffic.**

Monitoring tools helps network administrators in evaluating and diagnosing problems with servers, the network wire, hubs and applications. But other than artificial intelligence machines cannot recognize personalities and content, they can also be used for communication and exchange of information pertaining to unlawful activity. This is why law enforcement agencies have shown increased interest in network monitoring tools and application to analyse the data. By monitoring and analysing the data, flowing across the net can help detect and prevent crime. Such monitoring tools and applications have an important role in intelligence gathering.

In this paper explained related HTTP packet sniffing tools such as wireshark, TCP DUMP, TCP Flow, and Colasoft Capsa and their limitations of reconstructing HTTP web pages by packet analysis & decode. I gave a brief introduction of packet sniffing and how it is works and its components and Challenges to extract HTTP web pages data.

Keywords—**Networking, HTTP, Monitoring, Analyse, Sniffing.**

I. INTRODUCTION

Packet sniffing is the part of monitoring, capture of data traffic on a computer network. Data sent between computers over the internet or between any networks holding the form of small chunks called packets, which are routed to their target and assembled back into a complete message. Packet sniffer is a program running in a network attached Device that passively receives all data link layer frames, which are passing through the device's network adapter. It is also known as Network or Protocol Analyzer or Ethernet Sniffer [1].

The packet sniffer captures the data that is addressed to other machines, saving it for later analysis. Packet sniffers are mostly used by network to monitor trouble shoot the network traffic. The packet sniffing occurs by two ways either in promiscuous mode or monitor mode. In promiscuous mode, NIC of this system capable to receive over all packets in the form raw frame from network, namely this system (involving its software) is a sniffer. When a packet is received by a NIC, it first compares the MAC address of the packet to its own. If the MAC address matches, it accepts the packet otherwise filters it. This is because of the network card discarding all the packets that do not contain its own MAC address. An operation mode called no promiscuous, which basically means that each network card is minding its own business and they only reading the frames, which are directed to it. In order to captures the packets, NIC has to be set in the promiscuous mode. Packet sniffers will do sniffing by setting the NIC card of its own system to promiscuous mode, and hence captures all packets even they are not intended for it. So, packet sniffer captures the packets by setting the NIC card into promiscuous mode, when the packet arriving at the NIC, which are copied to the device driver memory, and then passed to the kernel buffer from where it is used by the user application.

Monitor mode (RFMON) enables a wireless NIC to capture packets without associating (means some authentication) with an access point. In monitor mode the NIC does not care whether the CRC(Cyclic Redundancy check) values are correct for packets captured in monitor mode, so some packets that you see may in fact be corrupted. For capture all traffic that the adapter can receive, the adapter must be put into "monitor mode", sometimes called "rfmon mode". In the monitor mode, the driver will put the adapter in a mode where it will supply to the host packets from *all* service sets.

II. SNIFFER COMPONENTS

Basic Components of sniffers are:-

A. *The hardware*: -
Most of the products work from standard network adapters, though some require special hardware. By use special hardware, you can analyze hardware faults like CRC errors, voltage problems, cable programs, "dribbles", "jitter", negotiation errors, and so forth.

B. *Capture driver:*-This is the very crucial part. It captures the network traffic data from the wire, filters it for the particular traffic you want, and then stores the data in a buffer.

C *Buffer:* - When the frames are captured from the network, they are stored in a buffer.

D *Analysis & Decode:* - This displays in the form descriptive text of the contents of network traffic, so that an analysis can figure out what is going on.

III. WORKING

In the network, when computer sends a data in the form of packets. These packets are the chunks of data are actually directed to the certain designated system. Every sent data has a predefined destination point. So, all the data are directly directed to a particular computer. Normally a

system in a network is designed to receive the packets and read only those data which are intended for it, the sniffing process involves a cooperative effort between software and hardware. Process can be broken down into three steps.

- Packet sniffer collects the packets in the form of raw binary from the wire. Normally, this is done in switched networks, if the selected network interface will be promiscuous mode.
- Then captured binary data is converted into a readable form.
- Analysis of the captured and converted data. After captured network data, the packet sniffer takes and verifies its protocol based on the information extracted, and then begins its analysis of that protocol's specific features [2].

IV. CHALLENGES TO EXTRACT HTTP WEB PAGES DATA FROM PACKETS

For extracting HTTP web pages content, the tools has to start from packets to TCP connections, from TCP connections to individual HTTP transactions, from individual HTTP transactions to HTTP requests and HTTP responses and then transferred data. That is the process involving extracting HTTP web pages data. This is most complex to implement correctly [3].

HTTP uses TCP as its underlying transport protocol causing the following issues:

- A TCP connection may be terminated at any point.
- On ending connection, some HTTP clients will close the connection via RST others will send a FIN signal as acknowledgement.
- Even in HTTP 1.0 one TCP connection can be used to transfer multiple HTTP requests and responses. That means TCP protocol is capable of handling more than one HTTP transmission.
- Within a single TCP connection, it is not possible to decide when a transfer has completed and when to start the new meta information, because there will be no indication of data transmission completed.
- Even an HTTP GET request may contain data.
  Demultiplexing of the TCP packets into HTTP transactions makes dealing with lost packets, retransmitted packets and reordered packets:
- Even containing the TCP open connection or close connection events, the packet sniffer may lose any packet. Therefore using a TCP connection as the demultiplexing unit is not reliable.
- The sniffer may lose the packet containing the HTTP header or response information and therefore has to ignore the data associated with the request.
- Even packets containing the newline dividing the HTTP response from the HTTP data can get lost, therefore it difficult to decide when the HTTP meta data ends and when the real data starts.
- If the packet containing the Web page data, but sometimes it may not always arrive at the sniffer location before the packet containing the HTTP response.

V. HTTP PACKET SNIFFING TOOLS AND LIMITATIONS

Several tools exist those can capture packets from network traffic, do some analysis & decoding , usually such tools will put the network card of a computer into promiscuous mode, this enables the computer to capture to the overall traffic on that section of the network. Filtering can be done based on the IP related header data present in the packets , usually such filtering represents simple criteria for the IP addresses and ports present in the packets. These passive network sniffing programs have been developed for either wired or wireless network measurement; the best-known are TCPDUMP and WIRESHARK and TCPFLOW and Colasoft Capsa. Each has certain limitations.

A. Wireshark:

On UNIX or Windows related systems, Wireshark (or Ethereal if referring to older versions) is open source software. It designed to provide network protocol analysing utilities to its users. Wireshark is a (GUI) graphical user interface network protocol analyser that provides a method to interactively browse packet data from a live network or for a previously saved capture file. Wireshark provides a number of functionalities: 1) filter captured data based on specified input, 2) organize packets together into complete TCP streams to allow easier analysis of specific connections, 3) read and write captured packets into a variety of formats for compatibility with other applications and 4) provide statistics, it capable to allow correlation between packets and provide results from analysing network protocols. These functionalities are extremely useful, but can still potentially create difficulties in identifying the actual content being viewed.

Wireshark is extremely useful in technically analysing data packets. It displays the data effectively and uniformly in byte format, displaying and identifying different byte fields in the data packets. However if a user is not used to examining how packet headers worked, formats of packets, or even the hexadecimal numbering systems used here, then this data may appear as hard to understand gibberish. Wireshark allows a user to assemble all data into continuous TCP connections or streams.

A TCP stream is a connection between two computers between (target element and server) where multiple transmissions take place. These streams may send one file or many files between two computers maintaining the TCP stream. To construct the TCP stream, the user selects a packet and that enables the “Follow TCP Stream” method. By following on the IP addresses and ports of each connection, Wireshark identifies all packets, which are that match the parameters of the selected packet to assemble related data packets together and assembles. The TCP stream format can be difficult to follow for new users. It consist the initial HTTP packets that identify each file and follows that with a textual representation of assembled data packets in the TCP stream [4].

Limitations: TCP stream of wireshark provides a lot of information to a technically oriented person, for example: operating systems types, web browser, content (HTML or JPEG), language and others. Unless the user possesses a rudimentary understanding of HTTP packet headers,

HTML language, CSS language and can understand what an image looks like in byte format, however, it provides little information about what the web page, and able to reconstruct partial web pages after analysis but it require so much of manual work.

B. Tcp Dump:

Tcpdump is an open source Unix/Linux compatible. In windows it is called WinDump. It is command line application designed to provide a dump or capture of data traffic going over the network. Tcpdump provide prints out a description of the contents of data packets captured on a network interface. This dump can be supplemented with certain types of network traffic or can be used to dump all network traffic at the time of its operation. In addition to capturing and filtering received packets, Tcpdump provides a number of functionalities: 1) read and write captured traffic to data files in Packet Capture (PCAP) format, 2) filter packets based on specified parameters, and 3) print limited or full data information from each packet based on provided parameters. Tcpdump provides basic options, which will be use for network analysis while outputting its data to the screen or to data files [5].

Tcpdump provides a very basic view of the network traffic. By default, it prints a summary of the packets that were captured on the wire, without necessarily storing the data that those packets contained. There are options to provide more detailed information.

It contain the information such as captured time, IP addresses in Domain Name System (DNS) address, and port numbers, etc... The information is useful because it can help to determine what computers were involved in connections, general protocols (IP) and the time of packet capture. However, it does not provide much useful information about what a user is specifically looking at and what that information looks like. When Tcpdump is used to directly output packets to the hard drive, there is much more detailed information provided in the file due to following the PCAP format; otherwise Tcpdump returns the packet headers and data.

Limitations: Tcpdump provides information that is more directed toward the technically trained user. It does not provide any method for a user to view a complete data file, whether text based or visually, in order to determine what the data content is. This can limit the usefulness when the intent of network analysis is to determine and show visually what the actual content of packets included.

C. Tcp Flow:

Tcpflow is an open source application. It designed to provide a capture of traffic going over the viewable network. It differs from Tcpdump in that it reconstructs the actual data streams and stores each flow in a separate file for later analysis. By extracting data from response packets from a given web server to the client it construct. Then the data written in to a file based on the IP addresses and ports that are specified in this given TCP data stream.

Tcpflow allows a user to alternatively look at a live capture of traffic or take previously captured data from other applications such as Wireshark or Tcpdump. The main requirement of captured data traffic is that it needs to be in the PCAP format; this allows Tcpflow to read and extract data from PCAP files and convert the packets into the corresponding TCP data streams.

However, Tcpflow does not take the next step of extracting the data into the specific files contained in the data stream; by this I mean the images, HTML files, CSS files or even saved documents are combined together into one single file instead of saved individually to the hard drive for easy access. The user can only look at this information in its byte or text format [6].

Limitations: Tcpflow's output consists of the HTTP header information containing generic information such as type (text/html), date, etc...before any data file. In addition to the header information it contains the re-assembled data, the HTML file, in a text format. This is difficult to view in a browser because it is combined with the header information and other data information included in this TCP stream.

D. Colasoft Capsa:

Colasoft Capsa is a Network Analyzer, and it is a must-have freeware for network administrators to monitor, troubleshoot and diagnose their network. It is designed for the purpose of personal and small business use. Capsa Network Analyser Free Edition is an easy-to-use for network monitoring and troubleshooting purposes. It performs 24/7network monitoring, real-time packet capturing, reliable network forensics, advanced protocol analysing and in-depth packet decoding.

Limitation: A limitation of Colasoft Capsa is that, it works only on windows platform and it does not show the total web page, shows as separate links for individual files. Capsa Free is a great combination of powerful monitoring, in-depth packet decoding, reliable network diagnosing, real-time alerting and thorough reporting ability, and it provides you innovative solutions to numerous network problems [7].

In this paper compared the packet sniffing tools characteristics depending on attributes like supported OS, open source, no of protocols supported, Supporting PCAP, user interface type, multiple interface at single instance feasibility, display protocol layers in OSI 7 layers, identify the abnormal packets and forged data, reconstruct tcp streams, decoding forms, and able to reconstruct the HTTP web page data. The comparison shown in table 1

Table1 shows that none of the packet sniffer tool leads all the parameters. The advantages and disadvantages would help to develop a new packet sniffer application which could hide all the disadvantages of the most used packet sniffers and could outperform them on quantitative and qualitative parameters [8].

Table 2 shows after summarizing the previous result to extract the best tool against the given property for network monitoring and analysis of packets.

In my point of view, Wireshark and Capsa are both very powerful and popular packet sniffing tools for monitoring and analysing. But Colasoft Capsa has more user friendly interface and the data in an extremely easy-to-read manner.

TABLE I CHARACTERSTIC COMPARISION OF WIRESHARK, TCP DUMP, TCP FLOW AND COLASOFT CAPSA

S.No	Characteristics				
	Property	**Wireshark**	**TCP Dump**	**TCP Flow**	**Colasoft Capsa**
1	Supported OS	Windows and Linux	Linux	Linux	Windows
2	Open Source	Yes	Yes	Yes	No
3	No. Of Protocols Supported	More than 1000	TCP/IP	TCP/IP	300
4	Libpcap Based	Yes	Yes	Yes	No
5	User Interface	GUI and CLI	CLI	CLI	GUI
6	Multiple interfaces at single instance	No	No	No	Yes
7	Display protocol in OSI 7 layers	Yes	No	No	Yes
8	Identify the abnormal packet	No (only creates warning)	No	No	Yes
9	Identify the packets with forged data	Yes	No	No	Yes
10	Reconstructed TCP communication	Yes(but not formatted)	No	Yes	Yes
11	Decode protocol form (Hex, ASCII, EBDIC)	Only Hex and ASCII	Only hex and ASCII	Only ASCII	Yes
12	Reconstruct HTTP web pages	No, Show actual traffic content individually	No	No, Show actual traffic content in individual files	No, Show Links for traffic content individually

TABLE 2 CONCLUSION

S.NO	Conclusion	
	Property	**Best Tool**
1	Supported OS	Wireshark
2	User Interface	Colasoft Capsa
3	No. Of Protocols Supported	Wireshark
4	Multiple interfaces at single instance	Colasoft Capsa
5	Display protocol in OSI 7 layers	Colasoft Capsa
6	Identify the abnormal packet	Colasoft Capsa
7	Identify the packets with forged data	Colasoft Capsa
8	Decode protocol form (Hex, ASCII, EBDIC)	Colasoft Capsa

VI. CONCLUSION

There are a number of tools that allowed a user to examine and analyse captured web traffic. Each of the tools that I identified allows the user to examine captured traffic at a low, technically oriented level by directly examining bytes or text formats. This may present a problem for network monitors or users to reconstruct the visual oriented web page. So we need to develop a software application to make it easier for monitors and users to look at and analyse and decode their captured network traffic.

REFERENCES

[1] S. Ansari, Rajeev S.G. and Chandrasekhar H.S, "Packet Sniffing: Brief Introduction", *IEEE Potentials, Dec 2002- Jan 2003, Volume: 21 Issue: 5, pp: 17 – 19.*

[2] BoYu"Based on the network sniffer implement network monitoring Computer Application and System Modeling(ICCASM), 2010 *International Conference on Volume:* **7**, 2010, Page(s): V7-1-V7-3(2010).

[3] B. A. Forouzan, "World Wide Web: HTTP," TCP/IP Protocol Suite, 3rd ed. New Yourk: McGraw Hill, 2006, ch. 22.

[4] Gerald Combs. (2008, Jan.). Wireshark Network Analyzer Man-pages. Wireshark.org. [Online]. Available: http://www.wireshark.org/docs/man- pages/wireshark.html

[5] V. Jacobson, C. Leres, and S. McCanne. (2008, Jan.). The Tcpdump Manual Page. Lawrence Berkely Laboratory, Berkeley, CA, Jun 1989.

[6] Elson, Jeremy. (2008, Jan.). The Tcpflow Manual Page. Circlemud.org [Online]. Available: http://www.circlemud.org/~jelson/software/tcpflow/tcpflow.1.html

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

[8] C. Gandhi, G. Suri, R. P. Golyan, P. Saxena, and B. K. Saxina, "Packet Sniffers- A Comparative Study", IJCNCS, paper 2308-9830, p. 179-187.