Proceedings of Informing Science & IT Education Conference (InSITE) 2015

Cite as: Awodele, O., Oluwabukola, O., Ogbonna, A. C., & Adebowale, A. (2015). Packet Sniffer – A comparative characteristic evaluation study. *Proceedings of Informing Science & IT Education Conference (InSITE) 2015,* 91-100. Retrieved from http://Proceedings.InformingScience.org/InSITE2015/InSITE15p091-100Awodele1537.pdf

Packet Sniffer – A Comparative Characteristic Evaluation Study

Oludele Awodele, Otusile Oluwabukola, A.C Ogbonna, and Ajayi Adebowale
Babcock University, Ilishan-Remo, Ogun State, Nigeria

delealways@yahoo.com buhkieotusile@yahoo.com
acogbonna06@yahoo.com deboxyl@yahoo.com

Abstract

As network continue to grow it is increasingly important that network administrators are aware of the different types of traffic that is traversing their networks as well as provides sufficient means for decision making process. Traffic monitoring and analysis is essential in order to more effectively troubleshoot and resolve issues when they occur, so as not to bring network services to a stand still for extended periods of time. Numerous tools are available to help administrators with the monitoring and analysis of network traffic.

This paper present a comparative analysis of five existing packet sniffers vis-à-vis their performance.

Keywords: Network Traffic Monitoring and Analysis Tools, Sniffer Traffic Flow, snoop, tcpdump, Ngrep, Wireshark.

Introduction

The success of the Internet brought networking to every house and company, and every day new applications and technologies are created. The power and complexity of computer networks are growing every day but makes the work of the network administrator difficult in trying to secure the network.

Network monitoring and measurement have become more and more important in a modern complicated network. In the past, administrators might only monitor a few network devices or less than a hundred computers with a lesser bandwidth, however, now administrators have to deal with not only higher speed wired network but also wireless networks. Administrators need to monitor traffic movement and performance throughout the network and verify that security breaches do not occur within the network therefore they need more sophisticated network traffic monitoring and analysis tools in order to maintain the network system stability and availability such as to fix network problems on time or to avoid network failure, to ensure the network security strength, and to make good decisions for network planning.

Network monitoring is a difficult and demanding task that is a vital part of a

Editor: Eli Cohen

Network Administrators job. Network Administrators are constantly striving to maintain smooth operation of their networks. For this reason there is an increasing need for tools that can analyse, diagnose and test the functionality and the security of networks, these tools in order to perform their work, need to obtain the data transiting on a network, capturing it while the network is working. The capture process consists of obtaining, listening on the network for every transiting frame, independently from its source or destination. The computer sending the data is known as the source and the computer receiving the data is the destination. To send data over the internet, the source computer must first break the data into smaller pieces known as packets.

Analysis Tools

Network analysis is the process of capturing network traffic and inspecting it closely to determine what is happening on the network."

There are various kinds of tools dealing with the network analysis. Given the data packet and network traffic flow information, administrators can understand network behaviour, such as application and network usage, utilization of network resources, and network anomalies.

Increasing computer networks increase the demand of network administrators which further increase the demands of packet sniffers. There are different types of passive packet sniffers used are: - Wireshark, TCPdump, Colasoft packet sniffer – Colasoft Capsa, etherdetect and ettercap, Ngrep, Snoop. In this section, key features of top 5 passive packet sniffers are discussed in detail i.e. Wireshark, TCPdump, Ngrep, Etherape and Snoop.

Tcpdump by McCanne, Leres and Jacobson (1987)

This is one of the most popular packet sniffers; it is accompanied by the libpcap library. It was originally written in 1987 at the Lawrence Berkeley National Laboratory and published a few years later and quickly gained users attention. Libcap is a C library for capturing packets.

The procedures included in libpcap provide a standardized interface to all common (UNIX based) operating systems, including Linux and FreeBSD. The interface of the libpcap is usable even under Windows but there the library is called winpcap.

So, it is useful to design applications to use libpcap for ensuring portability. It is an open source library that provides a high level interface to the network capture systems. Libpcap authors' main objective was to create a platform-independent API to eliminate the need for system-dependent packet capture modules in each application, as virtually every OS vendor implements it own capture mechanisms. The libpcap API is designed to be used from C and C++. However there are many wrappers that allow its use from languages like perl, java, C# or Ruby. Libpcap runs on most UNIX-like operating systems.

Tcpdump is a common packet analyser that runs under the command line and parsing tool ported to several platforms. It allows the user to intercept and display TCP/IP and other packets being transmitted or received over a network to which the computer is attached.

Tcpdump works by capturing and displaying packet headers and matching them against a set nof criteria. It runs on most UNIX-like operating systems - e.g. Linux, BSD, Solaris, Mac OS X, HP-UX and AIX amongst others making use of the libpcap library to capture packets. The user may optionally apply a BPF-based filter to limit the number of packets seen by tcpdump; this renders the output more usable on networks with a high volume of traffic. [Wikipedia 2011]

Since its inception, tcpdump was cited by numerous scientific papers in the field of computer networks and is indeed the standard utility for capturing IP traffic. It established an output file format PCAP, which is the most popular file format for storing IP packets off-line, still developed

to support new functionality. The tcpdump sniffer features a commandline filter mechanism, which allows the user to easily capture only the packets satisfying given criteria, e.g. TCP packets with destination port number equal to 80. Unfortunately, this filter mechanism does not support selecting packets of a single application only – especially if the monitored process is a peer-to-peer application, allocating new ports each few seconds. It prints out a description of the contents of packets on a network interface that match the boolean expression of the filter. It can also be run with the **-w** flag, which causes it to save the packet data to a file for later analysis, and/or with the **-r** flag, which causes it to read from a saved packet file rather than to read packets from a network interface. In all cases, only packets that match expression will be processed by tcpdump.

Tcpdump is frequently used to debug applications that generate or receive network traffic. It can also be used for debugging the network setup itself, by determining whether all necessary routing is occurring properly, allowing the user to further identify the source of a problem. It is also possible to use tcpdump for the specific purpose of intercepting and displaying the communications of another user or computer. It is a fairly simple and easy to use utility which can be used for any general packet monitoring mechanism in promiscuous mode.

Tcpdump can view the entire data portion of an Ethernet frame or other link layer protocol: An IP packet, an ARP packet, or any protocol at a higher layer than Ethernet. Tcpdump is very useful to a network administrator, because it gives a very clear picture of a specific part of your network; it can be used, when the problem is simply that something is not working properly, and helps debug denial of service attacks. Tcp show the source address, destination address, type of traffic, etc. and also checks the packet contents to learn more about the nature of the attack.

This approach has several limitations, because the data is being written directly to a file, you must know when to terminate recording without actually seeing the traffic. Also, if you limit what is captured with the original run, the data you exclude is lost. For these reasons, you will probably want to be very liberal in what you capture, offsetting some of the storage gains of the binary format. Clearly, each approach has its combination of advantages and disadvantages

The major drawbacks to tcpdump are the size of the flat file containing the text output and that tcpdump runs using the command line. Since tcpdump is text based, it is easy to run remotely using a Telnet connection. Its biggest disadvantage is a lack of analysis. However there are few other disadvantages with tcpdump. These include:

1. Limitation on the type of network traffic that can be analysed i.e. only TCP based protocols can be seen,

2. It is a fairly manual implementation with the user required to know all the options for screening specific packets. - No easy user interface and external implementations like user defined network condition analysis require use of other tools like tcppreplay or tcpopera alongside

3. Packets blocked by a gateway firewall may not be seen

Wireshark by Gerald Combs (1997)

In late 1997, Gerald Combs needed a tool for tracking down networking problems and wanted to learn more about networking, so he started writing Ethereal (the former name of the Wireshark project) as a way to solve both problems. Wireshark is an open source software project, and is released under the GNU General Public License (GPL).Wireshark currently runs on most UNIX platforms. The system requirements should be comparable to the Windows values listed above. Wireshark is a free and open- source packet analyser and it is written in C. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Originally named Ethereal, in May 2006 the project was renamed Wireshark due to trademark issues.

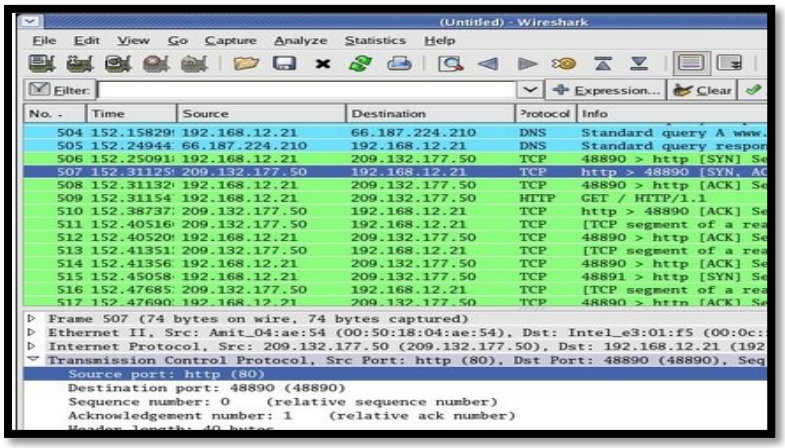
Wireshark is cross-platform, using the GTK+ widget toolkit to implement its user interface, and using pcap to capture packets; it runs on various UNIX-like operating systems including Linux, Mac OS X, Solaris, and on Microsoftt Windows. There is also a terminal-based (non- GUI) version called TShark. Wireshark, and the other programs distributed with it such as TShark, are free software, released under the terms of the GNU General Public license.

Wireshark is very similar to tcpdump, but has a graphical front-end, plus some integrated sorting and filtering options.

Wireshark allows the user to put network interface controllers that support promiscuous mode into that mode, in order to see all traffic visible on that interface, not just traffic addressed to one of the interface's configured addresses and broadcast/multicast traffic. However, when capturing with a packet analyser in promiscuous mode on a port on a network switch, not all of the traffic traveling through the switch will necessarily be sent to the port on which the capture is being done, so capturing in promiscuous mode will not necessarily be sufficient to see all traffic on the network. Port mirroring or various network taps extend capture to any point on net; simple passive taps are extremely resistant to malware tampering.

Wireshark is software that "understands" the structure of different networking protocols.

Thus, it is able to display the encapsulation and the fields along with their meanings of different packets specified by different networking protocols. Wireshark uses pcap to capture packets, so it can only capture the packets on the types of networks that pcap supports.



Display of Wireshark

Ngrep by Jordan Ritter (2001)

Ngrep strives to provide most of GNU grep's common features, applying them to the network layer. Ngrep is a pcap-aware tool that will allow you to specify extended regular or hexadecimal expressions to match against data payloads of packets. It currently recognizes IPv4/6, TCP, UDP, ICMPv4/6, IGMP and Raw across Ethernet, PPP, SLIP, FDDI, Token Ring and null interfaces, and understands BPF filter logic in the same fashion as more common packet sniffing tools, such as tcpdump and snoop. A "network grep" system is based around raw packet capture pumped through a "regular expression" parser that finds patterns in the network traffic. An example pattern would be: "/cgibin/ phf", which would indicate an attempt to exploit the vulnerable CGI script called "phf". Once building such a system, one can then analyse well-known attacks, extract strings specific to those attacks, and add them to database of patterns.

"Regexp" (regular expression) is a common pattern-matching language in the UNIX environment. While it has traditionally been used for searching text files, it can also be used for arbitrary binary data.

In truth, such systems have more flexible matching criteria, such as finding ports or matching TCP flags. "libpcap" (library for packet capture) is a common library available for UNIX systems that "sniffs" packets off a wire. Most UNIX-based intrusion detection systems (of any kind) use libpcap, though many also have optimized drivers for a small subset of platforms.

Ngrep may simply feed the output of libpcap (or tcpdump) into the regular expression parse, where the expressions come from a file on the disk. Some even simpler systems don't even use regular expressions and simply compare packets with well-known byte patterns. Ngrep has traditionally been used to debug plain text protocol interactions such as HTTP, SMTP, FTP, etc., to identify and analyse anomalous network communications such as those between worms, viruses and/or zombies, and to store, read and reprocess pcap dump files while looking for specific data patterns. On the other hand, it can be used to do the more mundane plain text credential collection as with HTTP Basic Authentication, FTP or POP3 authentication, and so forth. Like all useful tools, it can be used for good and bad.

By using ngrep creatively, one can detect anything from virus activity. It can be easily see that ngrep has an advantage over other protocol sniffing tools. Because ngrep does not have pre-conceived notion about what network traffic is supposed to look like, it can often detect attacks that other tools might miss. For example, if a company is running a POP3 server on a different port, it is likely that protocol sniffing tool will not recognize the traffic as POP3.

Therefore, any attacks against the port might go undetected. On the other hand, a network grep style system doesn't necessarily care about port numbers and will check for the same signatures regardless of ports. However ngrep also has some disadvantages like:

1. Ngrep based systems result in larger numbers of false positives. For e.g - alarms to go off if there is a match with the regular expression even when no real threat is present.

2. Pure packet sniffers are often able to run faster because a protocol decodes doesn't have to "search" a frame.

EtherApe by Juan and Riccardo (2000)

EtherApe was originally authored by Juan Toledo and Riccardo Ghetta and the first version of EtherApe was released in the year 2000. The most recent version of EtherApe, authored by Riccardo Ghetta, was released in 2011. EtherApe was written in Cobol (C) language, it runs on UNIX and all UNIX-like Operating systems, it was licensed by the GNU General Public License.

EtherApe is a packet sniffer/network traffic monitoring tool. EtherApe is free, open source software. It requires root privileges to run for security purpose and as such, there can be risks to the machine(s) running EtherApe when connected to the internet

In EtherApe, network traffic is displayed using a graphical interface. Each node represents a specific host. Links represent connections to hosts. Nodes and links are color coded to represent different protocols forming the various types of traffic on the network. Individual nodes and their connecting links grow and shrink in size with increases and decreases in network traffic.

Snoop by Sun Microsystems (2006)

Snoop is a very flexible command line packet sniffer included as part of Sun Microsystems' Solaris Operating system. It was developed by Sun Microsystems in 2006 and runs on Solaris operating system and it’s a packet analyser. It is pretty cable sniffer equal or better than TCPdump.

Snoop file format is different from PCAP and was defined in RFC 1761. Snoop displays packets in a single-line summary form or in verbose multi-line forms. It can display packets as soon as they are received or saved to a file. When snoop writes to an intermediate file, packet loss under busy trace conditions is unlikely.

Snoop itself can be used read and interpret the file e.g - print summary expended summary and full packets dumps. It has pretty powerful packet filtering engine

The snoop command can capture both IPv4 and IPv6 packets. It can display IPv6 headers, IPv6 extension headers, ICMPv6 headers, and neighbour discovery protocol data. By default, the snoop command displays both IPv4 and IPv6 packets. IPv6 traffic snoop capabilities are very similar to tcpdump and output formats are almost identical. Still there are some differences and for example the IDS Snort can read tcpdump binary files, but not snoop binary files. Another tool Ethereal's editcap program can be used for converting the snoop file to a tcpdump file. However, it is only packets for which the expression is true that are selected. If no expression is provided it is assumed to be true. Given a filter expression, snoop generates code for either the kernel packet filter or for its own internal filter. If capturing packets with the network interface, code for the kernel packet filter is generated. This filter is implemented as a streams module, upstream of the buffer module. The buffer module accumulates packets until it becomes full and passes the packets on to snoop. The kernel packet filter is very efficient, since it rejects unwanted packets in the kernel before they reach the packet buffer or snoop. The kernel packet filter has some limitations in its implementation; it is possible to construct filter expressions that it cannot handle. In this event, snoop tries to split the filter and do as much filtering in the kernel as possible. The remaining filtering is done by the packet filter for snoop. The -C flag can be used to view generated code for either the packet filter for the kernel or the packet filter for snoop. If packets are read from a capture file using the -i option, only the packet filter for snoop is used.

A filter expression consists of a series of one or more boolean primitives that may be combined with boolean operators (AND, OR, and NOT). Normal precedence rules for boolean operators apply. Order of evaluation of these operators may be controlled with parentheses. Since parentheses and other filter expression characters are known to the shell, it is often necessary to enclose the filter expression in quotes. There are two forms in which snoop can display Summary form – Only the application level protocol is displayed. For e.g - for an NFS packet only NFS information displayed. All underlying RPC, UDP, IP, and Ethernet frame information is suppressed Verbose form – Display everything As it can be seen, snoop is a more efficient and useful monitoring tool compared to raw TCPdump since it has a much better user interface and displaying capabilities, besides having all the advantages of TCPdump. The only disadvantage being that since it is tightly integrated within the Solaris kernel, its use is largely limited to Sun based systems. Though there have been hacks to port it to other kernels like FreeBSD and Linux, most of them have been fairly limited by number.

Characteristic Evaluation

To compare the above tools qualitatively we need to finalize some parameters. These can be OS support, disk usage, cost, number of protocols supported, etc. but this network packet sniffers will be determined based on the operating systems they run. These free packet sniffers can analyze network packets of all outgoing traffic, and analyze information from them. Table 1 below shows the camparison existing network sniffers.

Table 1: comparison of packet sniffers based on Operating Systems

Client	Windows	Mac OS X	Linux	BSD	Solaris	Other
AppTransaction Xpert	Yes	Version 3.5 capture agents on POWERPC only	GUI, plus version 3.5 capture agents	No	Version 3.5 capture agents on SPARC only	Version 3.5 capture agents on AIX, and PA-RISC HP-UX only
Cain and Abel	Yes	No	No	No	No	No
Capsa Free Edition	Yes	No	No	No	No	No
Carnivore	Yes	No	No	No	No	No
Clarified Analyer	Yes	Yes	Yes	No	No	?
Clusterpoint Network Traffic Surveillance System	Yes	Yes	Yes	Yes	No	Any virtual-machine compatible OS
CommView	Yes	No	No	No	No	No
dSniff	?	Yes	Yes	Yes	Yes	?
EtherApe	No	Yes	Yes	Yes	Yes	?
Ettercap	Yes	Yes	Yes	Yes	Yes	?
I/O Ninja	Yes	No	No	No	No	No
Kismet	Yes	Yes	Yes	Yes	?	?
LANMeter	No	No	No	No	No	Fluke proprietary hardware
Netsniff-ng	No	No	Yes	No	No	No
NetworkMiner free edition	Yes	No	No	No	No	No
NetworkMiner Professional	Yes	No	No	No	No	No
Ngrep	Yes	Yes	Yes	Yes	Yes	AIX, BeOS HP-UX, IRIX, Tru64 UNIX
Microsoft Network Monitor	Yes	No	No	No	No	No
Observer	Yes	No	No	No	No	No
OmniPeek (formerly AiroPeek, EtherPeek)	Yes	No	No	No	No	No

PacketView Pro	Yes	No	No	No	No	No
Pt360 Tool Suite	Yes	No	No	No	No	No
Sniffer Portable	Yes	No	No	No	No	No
Snoop	No	No	No	No	Yes	No
Tcpdump	Yes (Win-Dump)	Yes	Yes	Yes	Yes	AIX, HP-UX, IRIX, Tru64 UNIX
Wireshark (formerly Ethereal)	Yes	Yes	Yes	Yes	Yes	AIX, HP-UX, IRIX, Tru64 UNIX
Xplico	No	No	Yes	No	No	No
Justniffer	No	Yes	Yes	Yes	Yes	*?*

Conclusion

Tcpdump is a powerful tool that allows administrators sniff network packets and make some statistical analysis out of those dumps. The major drawback to tcpdump is the size of the flat file containing the text output, while the other weakness is that tcpdump runs under the command line. It is a command-line network sniffing and parsing tool ported to several platforms. Wireshark is similar to tcpdump, but with additional features such as graphical user interface and many advanced sorting and filtering options, although wireshark is easier to use than tcpdump, it still limits the size of target-analyzing files. wireshark cannot analyze more than two-day network activities in personal computers. To examine more than two-day activities, network managers must control wireshark by iterating capturing and analyzing processes periodically to avoid excessive system memory uses. TcpDump is very economical in terms of memory, because its installation file size is just 484 KB, it does not have a user friendly Graphical User Interface (GUI). So the user has to study those commands and get acquainted with the command prompt like screen. On the other hand Wireshark has a very good user friendly GUI. But its installation file size is 18 MB and after installation it will consume 81 MB in Windows and a hefty 449 MB in Linux and it is written in C programming language. So in terms of memory requirements, it is very expensive. However ngrep has some disadvantages like its based systems result in larger numbers of false positives and pure packet sniffers are often able to run faster because a protocol decodes doesn't have to "search" a frame. This limitation may play a key role in not choosing it for use.

Snoop is a more efficient and useful monitoring tool since it has a much better user interface and displaying capabilities. The only disadvantage being that since it is tightly integrated within the Solaris kernel, its use is largely limited to Sun based systems.

References

Ansari, S., Rajeev, S., & Chandrashekar, H. (2002). Packet sniffing: A brief introduction. *IEEE Potentials, 21*(5), 17-19).

Asrodia, P., & Patel, H. (2012). Network traffic analysis using packet sniffer. *International Journal of Engineering Research and Applications (IJERA), 2*(3), 854-856. Retrieved from www.ijera.com

Awodele, O., & Otusile, O. (2012). The design and implementation of psniffer model for network security, *International Journal of Electronics Communication and Computer Engineering (IJECCE), 3*(6).

Dabir, A., & Matrawy, A. (2007). Bottleneck analysis of traffic monitoring using Wireshark. *Proceedings of the 4th International Conference on Innovations in Information Technology*,2007, IEEE Innovations '07 (pp. 158 – 162)

Flor, N. V., & Guillory, K. (2011). Technology corner: Internet packet sniffers. *Journal of Digital Forensics, Security and Law*, *6*(1).

Fuentes, F., & Kar, D. (2005). Ethereal vs. Tcpdump: A comparitive study on packet sniffing tools for educational purpose. *Computer Journal of Computing Sciences in Colleges*, *20*(4), 169-176.

Niphadkar, S. (2006). Analysis of Packet Sniffers – TCPdump VS Ngrep VS Snoop. *International Journal of Computer Science & Communication, 8*(3).

Packet analyser. (2014). In Wikipedia. Retrieved from http://en.wikipedia.org/wiki/Packet_analyzer

Packet aniffer. (2014). In Wikipedia. Retrieved from http://en.wikipedia.org/wiki/Packet_sniffer

TcpDump. (2009). Overview of TcpDump. Retrieved from http://www.tcpdump.org/

Winpcap. (2009). Sniffers: Wincap. Retrieved from http://www.wipcap.org/download

Wireshark. (2009). Wireshark: Introduction. Retrieved from http://www.wireshark.org/

Yu, B. (2010). Based on the network sniffer implement network monitoring. *Computer Application and System Modeling (ICCASM),* 2010 International Conference (Vol. 7, pp V7-1 - V7-3).

Biographies

Oludele Awodele Ph.D is presently the head of the department of computer science & mathematics, Babcock University, Ilishan-Remo, Ogun State, Nigeria. His research areas are Software Engineering, Data Communication and Artificial Intelligence. He has published works in several journals of international repute. He can be contacted at delealways@yahoo.com.

Otusile Oluwabukola holds a B.Sc degree (Computer Technology) and an M.Sc degree in Computer science from Babcock University, currently doing her PhD programme in Babcock University. Her research interests include Network Management, and Information Systems Security. She can be contacted at buhkieotusile@yahoo.com.

A.C Ogbonna Ph.D is presently the dean of School of Computer Science and Engineering, Babcock University ilisan Remo Ogun State, Nigeria. He can be contacted at acogbonna@yahoo.com.

Ajayi Adebowale holds a B.Sc. degree in Mathematics (Computer Science) and an M.Sc degree in Computer science from University of Agriculture Abeokuta, Nigeria and Babcock University, Nigeria respectively. He is a Cisco Certified Network Associate and his research interests include Knowledge discovery in databases, Machine learning and Information security. He can be contacted at deboxyl@gmail.com