A Packet Sniffer (PSniffer) Application for Network Security in Java

Otusile Oluwabukola, Awodele Oludele, A.C Ogbonna, Ajeagbu Chigozirim, and Anyeahie Amarachi
Babcock University, Ilishan-Remo, Ogun State, Nigeria

Abstract

This paper presents the full implementation of the Psniffer application software that captures network data as well as provides sufficient means for the decision making process of an administrator. The aim of this application is to rewrite C language sniffer into Java, and also develop an application that consumes little memory on the hard disk. This work illustrates the requirement needed to design a new application; it was developed in Java and it consumes little memory on the hard disk. This application comprises of five independent modules that handles different tasks efficiently. This program can monitor network traffic, analyzes traffic patterns, identify and troubleshoot network problems. This application does not transmit any data onto the network, uses 1MB of the hard disk space, friendly GUI and it is very easy to install.

Keywords: network traffic, packets, packet capture, packet analyzer/sniffer.

Introduction

Packets in computer communications can be defined as a quantity of data of limited size. In Internet all traffic travels in the form of packets, the entire file downloads, Web page retrievals, email, all these Internet communications always occur in the form of packets. In the internet, packet is a formatted unit of data carried by a packet mode in computer network

This work develop a PSniffer (Awodele & Otusile, 2012) network analyzer that can capture network traffic and analyze it and allows user to take only the feature as needed with little memory usage for installation and store it in a file to use it later in his work, then this will reduce the memory that is used to store the data unlike available tools can only capture network traffic without analysis, while some require large memory size for installation. In addition, have a user-friendly control interface (Awodele & Otusile, 2012).

Network sniffing is a network layer attack consisting of capturing packets from the network transmitted by other computers and reading the data content in search of sensitive information like passwords, session tokens and confidential information. This could be done using tools called network sniffers; these tools collect packets on the network and, depending on the quality of the tool, analyze the collected data like protocol decoders or stream reassembling.

A packet analyzer sometimes called a network analyzer, protocol analyzer or sniffer or Ethernet sniffer or wireless sniffer (Spangler, 2003), is a computer program or a piece of computer hardware that can intercept and log traffic passing over part of a network (Chan, 2002).

Sniffer is used as an assistant of network management because of its monitoring and analyzing features which can help to troubleshoot network, detect intrusion, control traffic or supervise network contents.

Why the Use of A Network Sniffer

The information running through networks is a valuable source of evidence for network administrators to fish out intruders or anomalous connections. The need to capture this information has lead to the development of packet sniffers.

A number of research works exist in the development of packet sniffers. However, the search for the ideal packet sniffer continues. Psniffer will come with additional functionalities such as 3D pie charts, a GUI and with little memory requirements.

Psniffer when installed in a network will help monitor network traffic and keeps log of all connections to the network, which is then analyzed for the detection of suspicious activities.

Packet Sniffer Tools

Several tools exist that can monitor network traffic, usually such tools will put the network card of a computer into promiscuous mode, this enables the computer to listen to the entire traffic on that section of the network. Filtering of this packets can be done based on the IP related header data present in the packets, usually such filtering specifies 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.

Tcpdump by McCanne, Leres and Jacobson

It is one of the most popular packet sniffers. Tcpdump 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.

Libpcap 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.

Tcpdump is a common packet analyzer 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 of 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.

Wireshark by Gerald Combs

Wireshark is a free and open- source packet analyzer 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 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 analyzer 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.

Difference between Existing Packet Sniffer Software and Psniffer

Tcpdump is a command-line network sniffing and parsing tool ported to several platforms. Wireshark is similar to tcpdump, but with a graphical user interface and many advanced sorting and filtering options. TcpDump is very economical in terms of memory since its installation file size is just 484 KB. TcpDump 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.

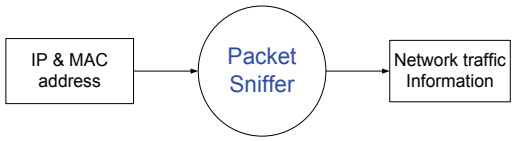
The limitation plays a key role in not choosing it for use. On the other hand Wireshark has a very good user friendly GUI, but its installation file size is 18 MB and after installation it will consume 81 MB in Windows and a hefty 449 MB in Linux. So in terms of memory requirements, it is very expensive.

The Psniffer is written in Java unlike the other Sniffers that are written in C language The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices. Java is a programmer's language that is cohesive and consistent, except for constraints imposed by the Internet environment, Java gives the programmer, full control. Finally, Java is to Internet programming where C was to system programming.

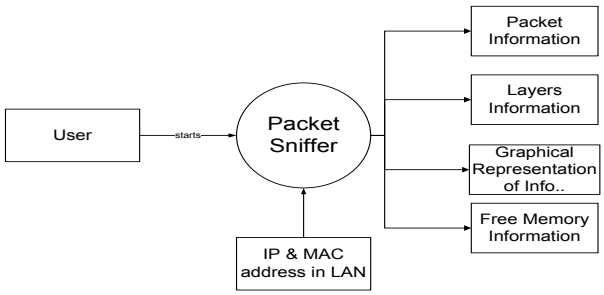
It captures packet, size of the packet, the source and destination machine IP addresses which are involved in the packet transferring. It shows this process in graphical manner and the working of different layers. It gives complete information about the captured packets; like which layers are involved and which protocols are in use at a particular time. Finally, it has a facility to store the information of the packets.

Dataflow Diagrams (DFDs)

DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system's structure charts. The Basic Notation used to create a DFD's are as follows:



Level-0 DFD



Level-1 DFD

Figure 1: Data flow diagram of the psniffer

Use Case Diagram

In software engineering, a use case diagram is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted in figure 2 below:

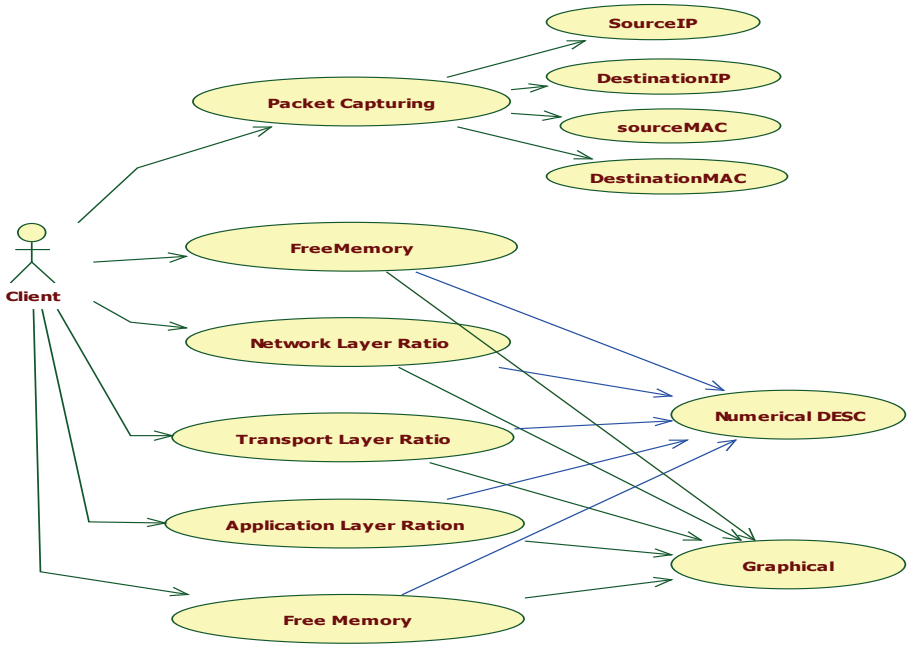


Figure 2: Use case diagram of the Psniffer

Component Diagram

A component diagram depicts how components are wired together to form larger software systems. Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component.

An assembly connector is a "connector between two components that defines that one component provides the services that another component requires. An assembly connector is a connector that is defined from a required interface or port to a provided interface or port."

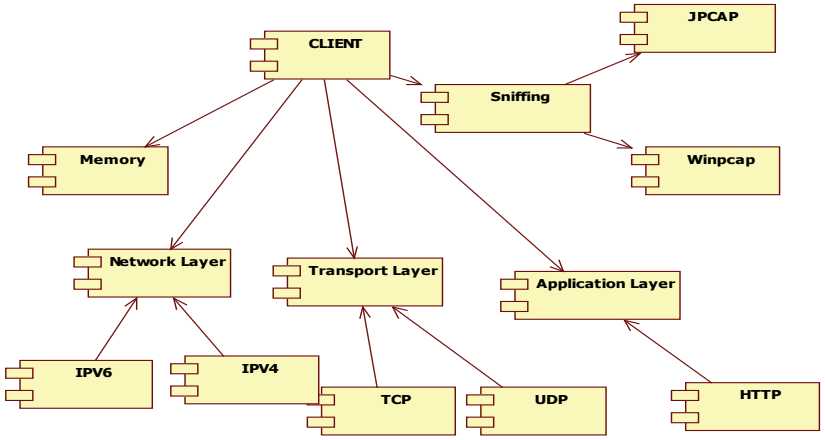


Figure 3: Component diagram of the Psniffer

Deployment Diagram:

A deployment diagram serves to model the physical deployment of artifacts on deployment targets. It shows "the allocation of Artifacts to Nodes according to the Deployments defined between them."

Deployment of an artifact to a node is indicated by placing the artifact inside the node. Instances of nodes (and devices and execution environments) are used in deployment diagrams to indicate multiplicity of these nodes. For example, multiple instances of an application server execution environment may be deployed inside a single device node to represent application server clustering.

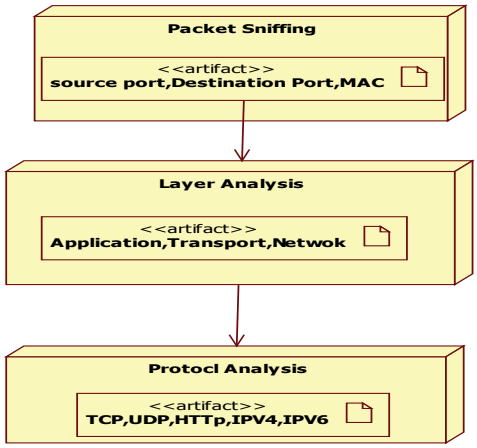


Figure 4: Deployment diagram of the Psniffer

Architecture of the Proposed System

The design of the proposed system discusses the various requirements that will make up the application. By conducting the requirements analysis we listed out the requirements that are useful to restate the problem definition.

- Analyze network Layer.
- Analyze Transport Layer.
- Analyze Application Layer.
- Analyze UDP Protocol
- Analyze TCP Protocol
- Analyze HTTP Protocol
- Analyze Free Memory Size
- Show Line-graph Representation.
- Show Pie chart Representation.
- Find out the Packets over network.

The Features of PSniffer

Psniffer is a customized software application that has a number of features. These features enable:

- Administrators to show statistics of received packets
- Administrators detect malicious IP addresses according to its number of ARP requests in previously specified time
- Administrators to view all network interfaces and enable them to capture data from that interface and consequently save captured packets.
- Administrators generate reports that aid effective and efficient decision making.

The Psniffer is developed in Java™. This application is designed into five (5) independent modules which take care of different tasks efficiently.

1. User Interface Module.
2. Packet Sniffing Module.
3. Analyze layers Module.
4. Free Memory Module.
5. Protocol Analysis Module.

User Interface Module

Actually every application has one user interface for accessing the entire application. The user interface for the Psniffer application is designed completely based on the end users. It provides an easy to use interface to the users. This user interface has an attractive look and provides ease of navigation. Technically, the swing is used in core java for preparing this user interface.

Packet Sniffing Module

This module takes care of capturing packets that are seen by a machine's network interface. It grabs all the packets that goes in and out of the Network Interface Card (NIC) of the machine on which the sniffer is installed. This means that, if the NIC is set to the promiscuous mode, then it will receive all the packets sent to the network.

Analyze Layers Module

This module contains the code for analyzing the layers in the system. Mostly in this module we have to discuss about three layers Transport layer, Application Layer, Network Layer. The module shows the graphical representation of the usage of different layers in packet capturing time. It can show the graph in two manners like line graph and pie graph.

Free Memory Module

This module analyzes computer memory usage at the time of packet capturing. It can show the memory size in number format as well as graphical representation.

Protocol Analysis Module

This module analyzes the protocols of the layers. Like Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Hypertext Transfer Protocol (HTTP) etc. It can show the source port, destination port and packet length of the system of each protocol.

Installation

Installation on Windows requires WinPcap software which can be downloaded from winPcap website. Jpcap is a set of Java classes which provide an interface and system for network packet capture; it is required for packet capture in Java and built upon Libpcap which is a packet capture library in C language. Java Runtime Environment (JRE) 5.0 or higher will also be required to run this Java application. JFreeChart is another java library required for rendering 3D pie chart for captured packet statistics. More space may be required to store the captured packets since the required space on hard disk for installation is less than 1MB.

Implementation

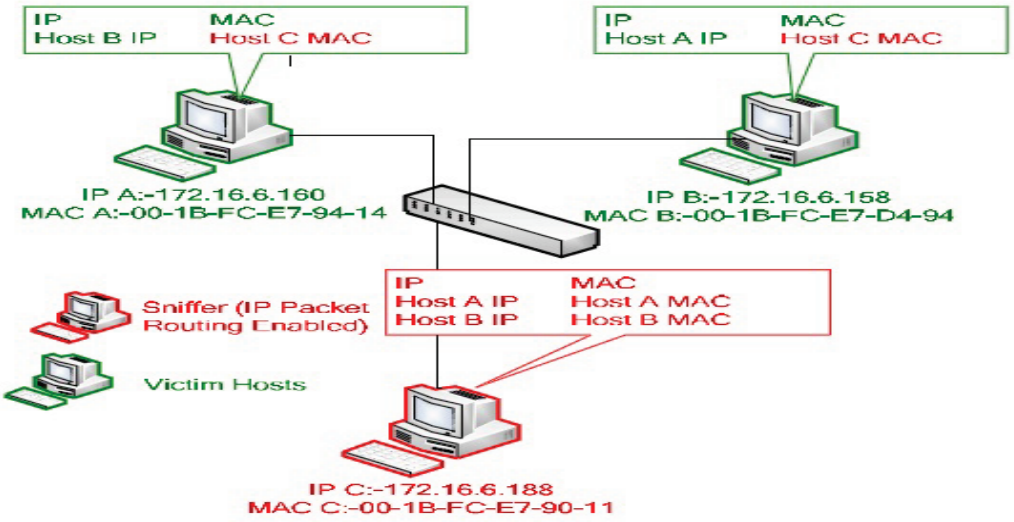


Figure 5: **ARP Cache poisoning that would be used for the implementation.**

Sniffers are programs that allow a host to capture any network packet illicitly. Specially, if the sniffers are active because active sniffer can alter or block network traffic while passive sniffer can only monitor network traffic. In this work the Passive Sniffer would be used.

There are two ways to sniff network traffic:

- • A host running a sniffer sets its NIC in promiscuous mode. If any host's NIC is running in promiscuous mode, it will receive all packets either those packets targeted to it or not. This way of sniffing is effective in an environment which is broadcast in nature like hub, access point and bus Local Area Network (LAN) environments.
- • ARP cache poisoning is also used for sniffing. This way of sniffing is effective in an environment, which is not broadcast in nature. Address Resolution Protocol (ARP) cache poisoning depends on local ARP cache maintained by each host of network. This cache contains IP with corresponding Media Access Control (MAC) addresses of recently accessed hosts.

Figure 5 explains ARP cache poisoning process that would be used for implementation. In this diagram, 'C' host performs ARP cache poisoning attack. 'C' host sends an ARP poison packet to target host 'A' which contains host 'C' MAC address in source MAC address field and host 'B' IP address in source IP address field of ARP poison packet. When target host 'A' receives this packet, it poisons local ARP cache value either by adding false entry or updating old entry with new one. Same process is repeated with host 'B'. This process corrupts the local ARP caches of host 'A' and 'B' which are shown in Figure 5. After the completion of poisoning process, both hosts cannot communicate directly with each other. Each host sends a packet to sniffer host and sniffer host reroutes packet back to actual destination. Sniffer host must have IP packet routing enabled so that it could send packet back to actual destination after getting confidential information.

Benefits of Psniffer

Psniffer has many benefits over the existing models. Listed below are the benefits.

- • It captures the live packet information in promiscuous and non-promiscuous mode.
- • It shows all the network interfaces and enables to capture data from that interface.
- • It also shows the statistics of the received packets.
- • It can save the captured packets.
- • It can retrieve the contents of the previously saved packet capture (Pcap) file.
- • It can show the TCP flow graph generated from the received TCP packets.

Interface of PSniffer

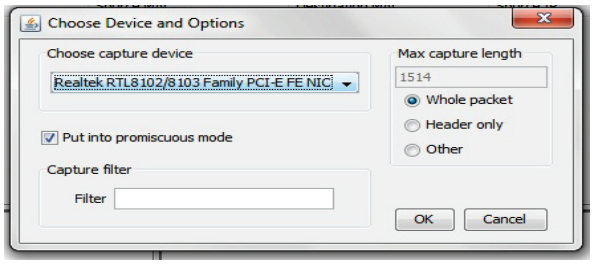


Figure 6: The main GUI of PSniffer

Figure 6 shows the user interface of the sniffer, where the device captured model will be determined either WAN or LAN.

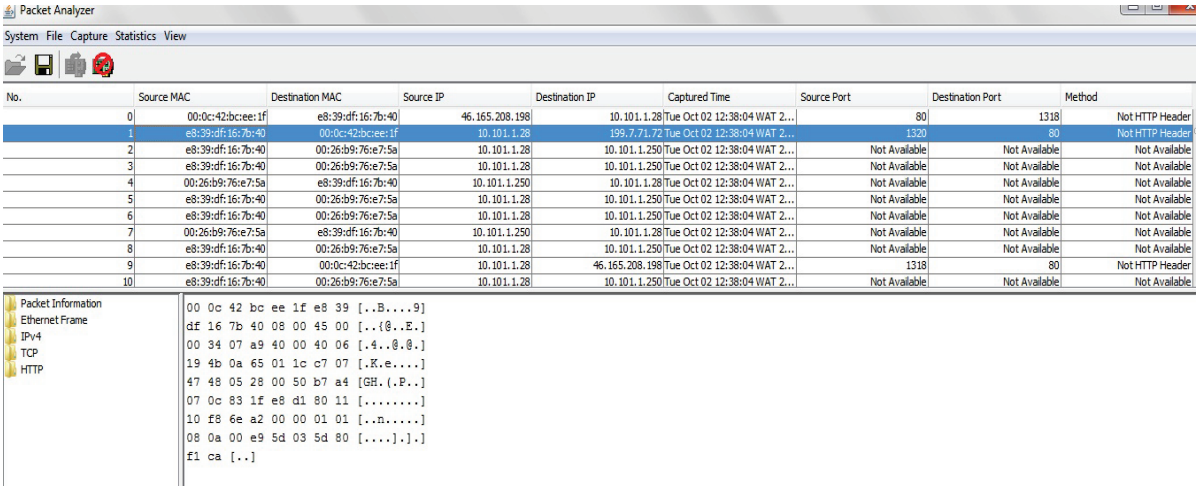


Figure 7: Captured packets containing necessary information

Figure 7 shows details of the captured packets showing the Source Mac and IP addresses, Destination Mac and IP addresses and methods of system on the network at the time it was sniffed.

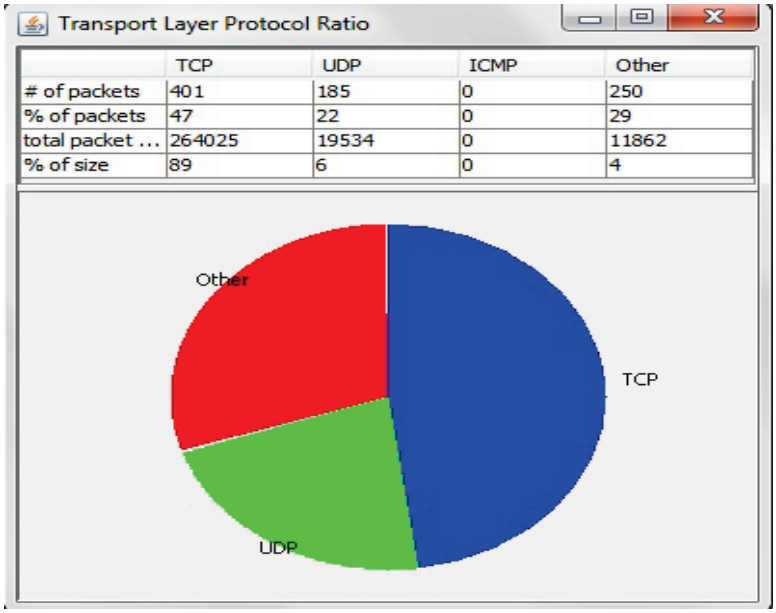


Figure 8: 3D pie chart showing received packet characteristics on transport layer

Figure 8 shows the pie chart (percentage, total packets and size) of protocols used on the transport layer at the time the network was sniffed.

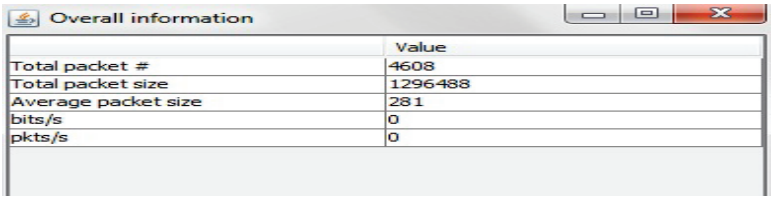


Figure 9: Overall information sent

Figure 9 shows the overall information of the packets sent over the network at the time it was sniffed.

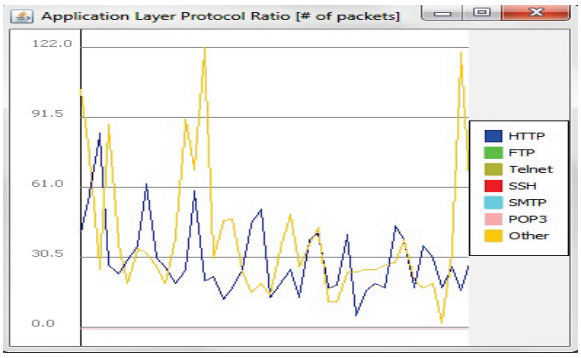


Figure 10: Graph showing the ratio of used supported application

The figure 10 shows the graph of the protocols (http, ftp, telnet) used on the application layer at the time it was sniffed.

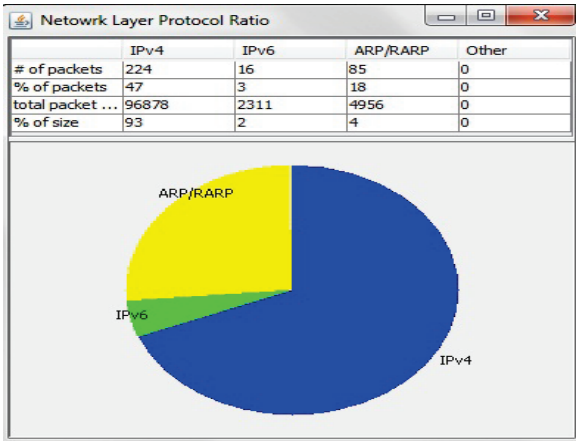


Figure 11: 3D pie chart showing received packet characteristics on Network Layer

The figure 11 show the Pie Chart (percentage, total packets) of the internet protocol type used on the network at the time the network was sniffed.

Conclusion

Compared to similar works this application shows the layer involved in sniffing and the protocols. This Passive Sniffer would be installed on a collision domain that makes use of the switch rather than the broadcast domain (HUB). The collision domain would be used since the use of HUBS in network setting is gradually reducing due to its broadcasting nature.

Psniffer has a very rich and user friendly GUI developed in Java Swing Technology. Thus it is totally easy to use. With Java, the most considerable advantage is platform independence; therefore Psniffer is also platform independent. The installation file for Psniffer is only 587 KB, so it is highly economical in terms of memory use and because it is based on object-oriented design, any further changes can be easily adaptable.

Future Enhancements

It is not possible to develop a system that meets all the requirements of the user. User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system are:

- As the technology emerges, it is possible to upgrade the system that can be adaptable to desired environment.
- The present application is a standalone application, i.e. only in intranet. So we have chance to extend this in internet.
- Based on the future security issues, security can be improved using emerging technologies.

Additional Sources

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)* [www.ijera.com], *2*(3), 854-856.

Brozycki, J. (2010). *Capturing and analyzing packets with Perl.*

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

Deri, L. (n.d.). *Improving passive packet capture: Beyond device polling*. Retrieved from http://www.net-security.org/dl/articles/Ring.pdf

Dhar, S. (2002). *Switchsniff.* Retrieved from http://www.linuxjournal.com/article.php

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

JFreeChart. (n.d.). *JFreeChart*. Retrieved from http://www.jfree.org/jfreechart/download.html

Jpcap. (2011). *Jpcap*. Retrieved from http://jpcap.sourceforge.net/

JRE. (n.d.). JRE. Retrieved from http://www.oracle.com/technetwork/java/javase/downloads/index.html

Kjell, B. (n.d.). *Introduction to computer science using Java*.

Lewis, J., & Loftus, W. (2001). *Java software solutions*. Addison Wesley.

McCanne, S., & Jacobson, V. (1992). *The BSD packet filter: A new architecture for user-level packet capture*.

Muna, M., Jawhar, T., & Mehrotra, M. (2010). System design for packet sniffer using NDIS hooking. *International Journal of Computer Science & Communication, 1*(1), 171-173.

Niphadkar, S. (2006). *Analysis of packet sniffers – TCPdump VS Ngrep VS Snoop*

Parmar, R., & Patel, H. (2011). NetCap: A packet sniffer in Java. *International Journal of Computer Science and Technology, 2*(3).

Senthil, K. P., & Arumugam, S. (2012). Establishing a valuable method of packet capture and packet analyzer tools in firewall. *International Journal of Research Studies in Computing, 1*(1), 11-20

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

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

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

References

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

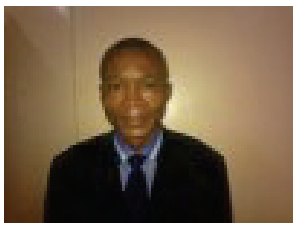
Chan, C. Y. (2002). *A network packet analyzer with database support*. Retrieved from http://www.cs.rpi.edu/~szymansk/theses/chan.ms.02.pdf

Spangler, R. (2003). Packet sniffer detection with antisniff. Retrieved from http://www.linux-sec.net/Sniffer.Detectors/snifferdetection.pdf

Biographies

Otusile Oluwabukola Received a B.Sc degree in Computer Technology from Babcock University 2009, and currently awaiting M.Sc degree in Computer science from Babcock University 2013. Her current research interests include Network Management, and Information Systems Security. She can be contacted at buhkieotusile@yahoo.com.

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.



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.

Ajaegbu Chigozirim received his M.Sc in Networking and Telecommunication from Babcock University. He is currently studying for a Doctorate degree in Telecommunication at Babcock University, Nigeria. His research interests include Cloud Computing, Network Management. He can be contacted at chigozirim.ajaegbu@yahoo.com

Anyaehie Amarachi received her B.Sc. in Computer Systems and Information Technology from Eastern Mediterranean University, North Cyprus, in 2011. She is currently studying for an M.Sc. degree in Computer Science at Babcock University, Nigeria. Her current research interests include Project Management, Web application development. She can be contacted at amarachi.anyaehie@yahoo.co.uk