Volume 2, Issue 11, November 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Enterprise Network Traffic Monitoring, Analysis, **and** Reporting Using WINPCAP Tool **With** JPCAP API

Navneet Kaur Dhillon and Mrs. Uzma Ansari
Dept. of CSE,
C.S.V.T.U
India,

Abstract: Today's enterprise networks are composed of multiple types of interconnected networks. Operations and management staff must provide an efficient, reliable and secure operating environment to support an organization's daily activities. Enterprise networks must be monitored for performance, security, and fault management .Current management uses the complex, hard-to-learn and hard-to-use tools. We need a simple, uniform, easy tools for managing networks. Client server based management promises to provide such solutions. This paper focuses on the use of packet capturing technology like WinPcap and JPCAP for the purposes of enterprise network traffic monitoring and reporting. In this paper, we first examine the requirements for enterprise network traffic monitoring, analysis and reporting, and then present the design and implementation of a network traffic monitoring and reporting system. Network traffic Monitoring is a network diagnostic tool that monitors local area networks and provides a graphical display of network statistics.While collecting information from the network's data stream, Network Monitor displays the following types of information:

- ***The source address of the computer that sent a frame onto the network.***
- ***The destination address of the computer that received the frame.***
- ***The protocols used to send the frame.***

The process by which Network Monitor collects this information is called capturing. By default, Network Monitor gathers statistics on all the frames it detects on the network into a capture buffer, which is a reserved storage area in memory. To capture statistics on only a specific subset of frames, we can single out these frames by designing a capture filter. To use Network Monitor, our computer must have a network card that supports promiscuous mode.

Keywords:- JPCAP. Libpcap , Network traffic , Packet capture , Winpcap .

1 INTRODUCTION

Enterprise networks are corporate computer networks composed of network devices, systems, and services supporting various corporate applications. Enterprise networks are growing rapidly in size and complexity because more systems and sophisticated applications run on them. The most challenging tasks facing network and system administrative staff today is to provide an efficient, reliable and secure computing environment. Administrators today typically use a management tool to monitor and control their networks. A management tool that is easy to learn and operate in a short period of time is desperately needed.

Enterprise networks are often large, run a wide variety of applications and protocols, and typically operate under strict reliability and security constraints; thus, they represent a challenging environment for network management. Indeed, most networks today require substantial manual configuration by trained operators to achieve even moderate security. Networks are most easily managed in terms of the entities we seek to control—such as users, hosts, and access points rather than in terms of low-level and often dynamically-allocated addresses. For example, it is convenient to declare which services a user is allowed to use and to which machines they can connect. Network policies dictate the nature of connectivity between communicating entities and therefore naturally affect the paths that packets take. This is in contrast to today's networks in which forwarding and filtering use different mechanisms rather than a single integrated approach. A policy might require packets to pass through an intermediate middlebox. Traffic can receive more appropriate service if its path is controlled directing real-time communications over lightly loaded paths, important communications over redundant paths, and private communications over paths inside a trusted boundary would all lead to better service. Today, it is notoriously difficult to reliably determine the origin of a packet: Addresses are dynamic and change frequently, and they are easily spoofed. The loose binding between users and their traffic is a constant target for attacks in enterprise networks. This requires a strong binding between a user, and the machine they are using, and the addresses in the packets they generate.

The purpose of this paper is to provide efficient and reliable secure enterprise network which will lead to the protected working environment

2 REQUIREMENTS

In this section we examine some of the most important requirements needed for a client-server based enterprise network monitoring and reporting system. Enterprise networks typically consist of network devices such as routers, bridges, switches, hubs and so on. A network management system must be able to automatically detect all the network devices in an enterprise network. Although the topologies of enterprise networks do not change frequently, they do change occasionally. When the topology changes, the network management system must detect the change and modify the necessary information for proper monitoring and control.

The major requirements of this project are

1) A LAN connection in an enterprise network (IEEE 802.3 ethernet connection).
2) WinPcap packet capturing library for windows.
3) JPCAP Java Packet capture java library.

2.1 IEEE 802.3 Ethernet

2.1.1 Definition

Ethernet is a family of computer networking technologies for local area networks (LANs) commercially introduced in 1980. Standardized in **IEEE 802.3[5]**, Ethernet has largely replaced competing wired LAN technologies. Systems communicating over Ethernet divide a stream of data into individual packets called frames. Each frame contains source and destination addresses and error-checking data so that damaged data can be detected and re-transmitted. The standards define several wiring and signaling variants. Data rates were periodically increased from the original 10 megabits per second, to 100 gigabits per second.

2.1.2 Ethernet frames

A data packet on the wire is called a frame. A frame begins with preamble and start frame delimiter, followed by an Ethernet header featuring source and destination MAC addresses. The middle section of the frame consists of payload data including any headers for other protocols (e.g., Internet Protocol) carried in the frame. The frame ends with a 32-bit cyclic redundancy check, which is used to detect corruption of data in transit.

2.2 Windows packet capture (WinPcap)

WinPcap[3] is the industry-standard tool for link-layer network access in Windows environments it allows applications to capture and transmit network packets bypassing the protocol stack, and has additional useful features, including kernel-level packet filtering, a network statistics engine and support for remote packet capture. WinPcap consists of a driver, that extends the operating system to provide low-level network access, and a library that is used to easily access the low-level network layers. This library also contains the Windows version of the well known libpcap Unix API.

WinPcap is the packet capture and filtering engine of many open source and commercial network tools, including protocol analyzers, network monitors, network intrusion detection systems[1], sniffers, traffic generators and network testers. Some of these tools, like (Wireshark, Nmap, Snort, ntop)[2] are known and used throughout the networking community

Winpcap.org[3] is also the home of WinDump[3] , the Windows version of the popular tcpdump[4] tool. WinDump can be used to watch, diagnose and save to disk network traffic according to various complex rules. In the field of computer network administration, pcap (packet capture) consists of an application programming interface (API) for capturing network traffic. Unix-like systems implement pcap in the libpcap[9] library; Windows uses a port of libpcap known as WinPcap.

WinPcap consists of:

- x86 and x86-64 drivers for the Windows NT family (Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows 7, etc.), which use NDIS (Network Driver Interface Specification) to read packets directly from a network adapter;
- implementations of a lower-level library for the listed operating systems, to communicate with those drivers;
- a port of libpcap that uses the API offered by the low-level library implementations.

2.3 Java packet capture (JPCAP)

JPCAP[14] is based on libpcap/Winpcap, and is implemented in C and Java JPCAP is a Java library for capturing and sending network packets. Using JPCAP, we can develop applications to capture packets from a network interface and visualize/analyze them in Java. JPCAP can capture Ethernet, IPv4, IPv6, ARP/RARP, TCP, UDP, and ICMPv4 packets[15].

JPCAP is a set of Java classes that provide an interface and system for network packet capture and also a protocol library and tool for visualizing network traffic is included. JPCAP hides the low-level details of network packet capture by abstracting many network packet types and protocols into Java classes. Internally, JPCAP implements bindings to the libpcap system library through JNI (the Java Native Interface).

JPCAP is an open source library for capturing and sending network packets from Java applications. It provides facilities to:

- capture raw packets live from the wire.
- save captured packets to an offline file, and read captured packets from an offline file.
- automatically identify packet types and generate corresponding Java objects (for Ethernet, IPv4, IPv6, ARP/RARP, TCP, UDP, and ICMPv4 packets).
- filter the packets according to user-specified rules before dispatching them to the application.
- send raw packets to the network

When we want to capture packets from a network, the first thing we have to do is to obtain the list of network interfaces on machine. To do so, JPCAP provides JPCAPCaptor.getDeviceList()method. It returns an array of NetworkInterfaceobjects.

A NetworkInterfaceobject contains some information about the corresponding network interface, such as its name, description, IP and MAC addresses, and datatlink name and description.

The following sample code obtains the list of network interfaces and prints out their information.

```
//Obtain the list of network interfaces
NetworkInterface[] devices = JPCAPCaptor.getDeviceList();

//for each network interface
for (int i = 0; i < devices.length; i++) {
  //print out its name and description
  System.out.println(i+": "+devices[i].name + "(" + devices[i].description+")");

  //print out its datalink name and description
  System.out.println(" datalink: "+devices[i].datalink_name + "(" +
devices[i].datalink_description+")");

  //print out its MAC address
  System.out.print(" MAC address:");
  for (byte b : devices[i].mac_address)
    System.out.print(Integer.toHexString(b&0xff) + ":");
  System.out.println();

  //print out its IP address, subnet mask and broadcast address
  for (NetworkInterfaceAddress a : devices[i].addresses)
    System.out.println(" address:"+a.address + " " + a.subnet + " "+
a.broadcast);}
```

Once we obtain the list of network interfaces and choose which network interface to captuer packets from, you can open the interface by using JPCAPCaptor.openDevice()method. The following piece of code illustrates how to open an network interface. The next following steps are capture packets from the network interface, this can be done by two methods using a callback method or capturing packets one by one the next step is to set capturing filter which is followed by saving a captured packets into a file and the saved packets can be later read from a file. This is over all working of JPCAP through which we develop our API for capturing packets from a network interface.

3 INTRODUCTION TO WINPCAP

The software architecture for packet capture and network analysis for Win32 platforms we intend to describe in this paper contains a library **(*WinPCap*)[8]**, which adds to the operating system the ability to capture efficiently the traffic from different kinds of networks using the network adapter of the machine. The systems based on Unix platform contain a kernel component used as a capture driver named ***BPF*** **(Berkeley Packet Filter)[8]**. This part of the kernel is the base of the ***libpcap*** **[9]** library. Unlike Unix platforms, Windows operating systems haven't such a component. The implementation of a similar library used to capture data is more difficult to realize because it calls for an interaction with the networking part of the kernel. This is more difficult in the "Microsoft world" where kernel source code is not available.

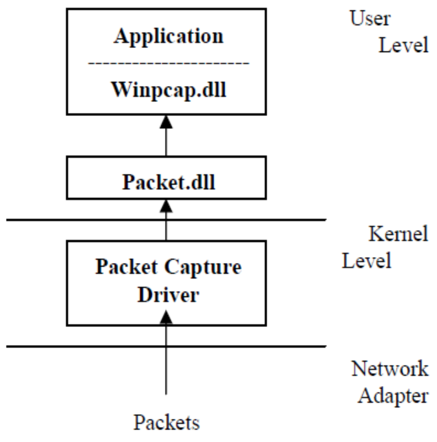


Figure :-1 Logical Structure of WinPcap

3.1 WinPCap Structure:-

Tocapture data from the network, a capture application must directly interact with the network adapter. Therefore the operating system must offer a set of primitives for capture with a view to communicate with the adapter. The purpose of these primitives is to capture network packets transparently from the point of view of the user and to transfer them to the calling application.

This part of the kernel should be quick and efficient in order to capture all the packets in real time without losing information.

The logical architecture of Winpcap is a hierarchical structure on 3 levels (from the network adapter to an application), shown in figure 1.

The ***Packet Capture Driver*** is the lowest software level of the capture structure. It is a part of the kernel and interacts with the network adapter to obtain captured packets. It supplies the application a set of functions used to read /write data from the network at data-link level.

Packet.dll**[12]** works at user level, but it is detached from the capture application. It is a dynamic link library that separates the application from capture driver providing a system-independent capture interface. It allows user's application to be executed on different Windows operating systems without being recompiled. It represents a set of API functions used to access the capture driver directly.

WinPCap.dll**[13]** (the third level) is a *static* library that is used by the packet captures part of the application. It uses the services exported by Packet.dll, and provides the applications a higher level and a powerful capture interface. It is statically linked; it means it is part of the application that uses it.

The user interface is the highest part of the capture application. It manages the interaction with the user and displays the result of a capture.

3.3 The working of a WinPCap

The WinPCap action is based on packets capture at the network adapter level. Therefore, any application using WinPCap must follow the steps summarized below:

- Specify the network adapter which will be used for capture
- Initialize winpcap (mention the network adapter if the capture is on-line)
- Offer a pattern for the captured data (a TCP/IP structure for example), so it can be done a cast at this pattern and obtain significant information (complex applications offer patterns for majority protocols)
  Analyze captured data according with the application type (captured data can be saved and processed off-line)
- Close the capture session

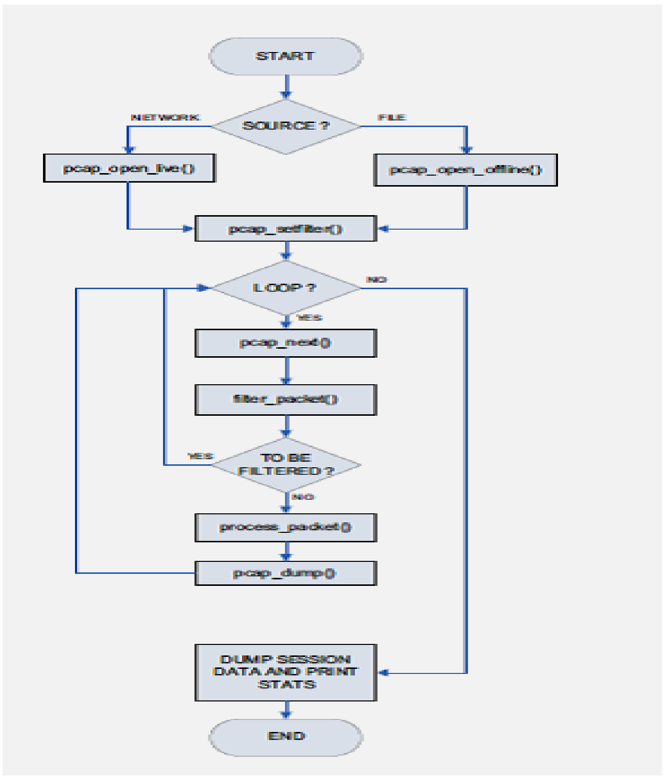


Figure:- 2 Flow Chart describing the working of Winpcap

The main functions performed by winPcap are :-
The first step in using the WinPCap Library is to select a network adapter. This adapter can be specified in two ways. The first way is by adding its name as argument in the command line (this works with a trivial application).

```
int main(int argc, char *argv[])
{
char *dev = argv[1];
printf("Network Adapter: %s\n", dev);
return(0);
}
```

The second technique uses the function ***pcap_lookupdev()*** to extract the list of all the adapters presented on current machine:

```
int main()
{
char *dev, errbuf[PCAP_ERRBUF_SIZE];
dev = pcap_lookupdev(errbuf);
printf("Network Adapters:%s\n", dev);
return(0);
}
```

pcap_open_live() this WinPCap function opens a capture session. It returns a handler to the current session and it is used for on-line capture. A similar function is used to retrieve saved data***pcap_ open_offline()***.
The ***pcap_open_live()*** function specifies the network adapter(dev) and sets it to normal/ promiscuous mode .
An important feature of WinPCap Library is that it can filter the traffic.
To filter the traffic, WinPCap Library offers two functions: ***pcap_compile()*** and ***pcap_setfilter()***.
The WinPCap Library contains the function ***pcap_lookupnet()*** which retrieves IP address and mask of a specified network adapter.
There are two techniques for capture data. The first technique uses the function ***pcap_next()*** to capture only one packet. The second technique uses a loop to capture all packets received this time. ***pcap_dispatch()*** or ***pcap_loop()***.

The call ***pcap_loop()*** is used in simple applications, while ***pcap_dispatch()*** is normally preferred in a more complex program.
In the ***process_packet()*** function, we could get the size of TCP payload one of our sampled parameters.

The working of WinPcap is shown in figure 2.

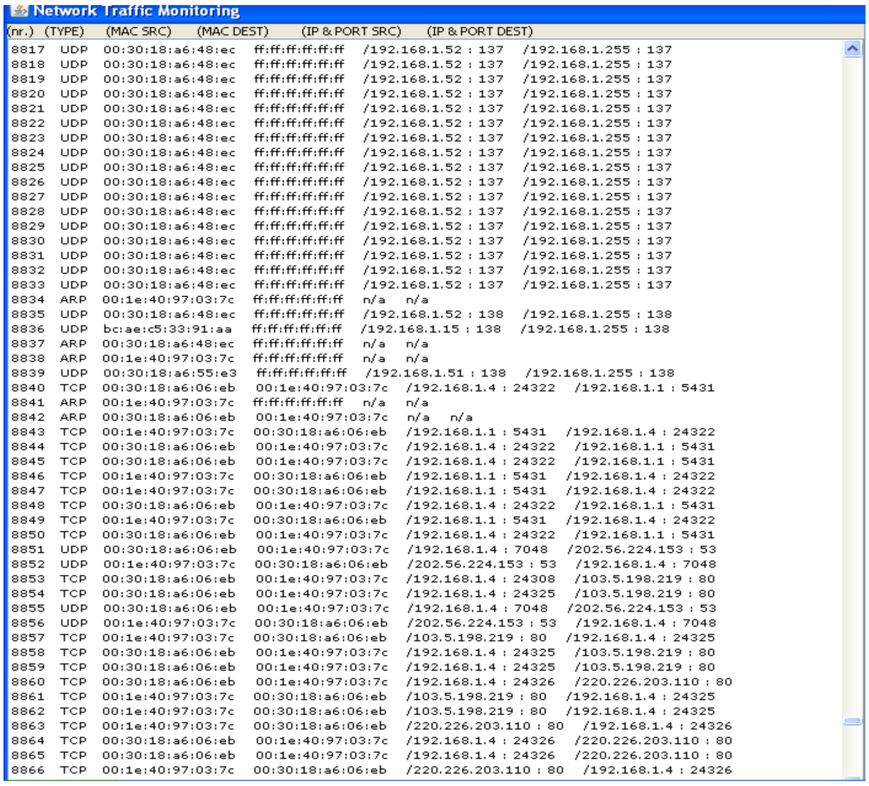
3.4 The Network traffic monitoring Application

The network traffic monitoring application is an interactive tool which uses the WinPCap architecture and C/C++ calls for packet captures. Although we used for its interface the MFC (Microsoft Foundation Class) functions, the network traffic monitoring has a certain degree of independence from the OS, by using the "compatible" calls of WinPCap. Use of threads reduces somehow the predictability of application behaviour only to systems similar to Windows NT, 2000 or XP.

The tool is just a preliminary step; it achieves the capture of all packets in the network and the specification of their type. For "standard" packet types like TCP/IP and UDP the component fields of the headers are specified, shaped as a tree structure.
A snapshot obtained during a capture session is shown in figure 3.

3.5 Report Generation

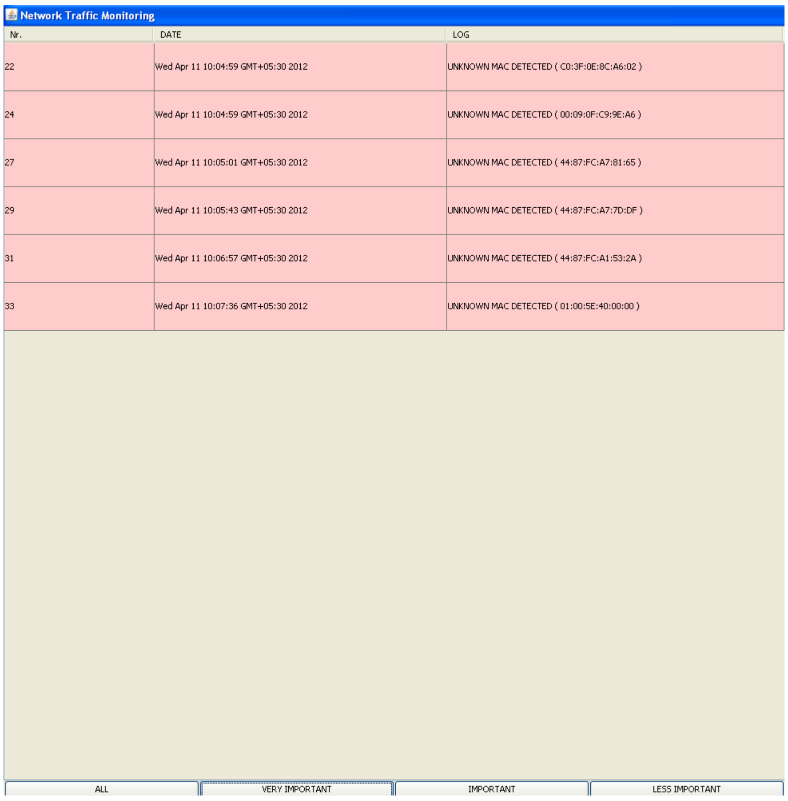
1) Traffic monitoring graphs.
2) Reports generating detection of spoofed packets



```
Network Traffic Monitoring
(nr.)  (TYPE)   (MAC SRC)        (MAC DEST)        (IP & PORT SRC)        (IP & PORT DEST)
8817  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 137  /192.168.1.255 : 137
8818  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 137  /192.168.1.255 : 137
8819  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 137  /192.168.1.255 : 137
8820  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 137  /192.168.1.255 : 137
8821  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 137  /192.168.1.255 : 137
8822  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 137  /192.168.1.255 : 137
8823  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 137  /192.168.1.255 : 137
8824  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 137  /192.168.1.255 : 137
8825  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 137  /192.168.1.255 : 137
8826  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 137  /192.168.1.255 : 137
8827  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 137  /192.168.1.255 : 137
8828  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 137  /192.168.1.255 : 137
8829  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 137  /192.168.1.255 : 137
8830  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 137  /192.168.1.255 : 137
8831  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 137  /192.168.1.255 : 137
8832  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 137  /192.168.1.255 : 137
8833  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 137  /192.168.1.255 : 137
8834  ARP  00:1e:40:97:03:7c  ff:ff:ff:ff:ff:ff  n/a   n/a
8835  UDP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  /192.168.1.52 : 138  /192.168.1.255 : 138
8836  UDP  bc:ae:c5:33:91:aa  ff:ff:ff:ff:ff:ff  /192.168.1.15 : 138  /192.168.1.255 : 138
8837  ARP  00:30:18:a6:48:ec  ff:ff:ff:ff:ff:ff  n/a   n/a
8838  ARP  00:1e:40:97:03:7c  ff:ff:ff:ff:ff:ff  n/a   n/a
8839  UDP  00:30:18:a6:55:e3  ff:ff:ff:ff:ff:ff  /192.168.1.51 : 138  /192.168.1.255 : 138
8840  TCP  00:30:18:a6:06:eb  00:1e:40:97:03:7c  /192.168.1.4 : 24322  /192.168.1.1 : 5431
8841  ARP  00:1e:40:97:03:7c  ff:ff:ff:ff:ff:ff  n/a   n/a
8842  ARP  00:30:18:a6:06:eb  00:1e:40:97:03:7c  n/a   n/a
8843  TCP  00:1e:40:97:03:7c  00:30:18:a6:06:eb  /192.168.1.1 : 5431  /192.168.1.4 : 24322
8844  TCP  00:30:18:a6:06:eb  00:1e:40:97:03:7c  /192.168.1.4 : 24322  /192.168.1.1 : 5431
8845  TCP  00:30:18:a6:06:eb  00:1e:40:97:03:7c  /192.168.1.4 : 24322  /192.168.1.1 : 5431
8846  TCP  00:1e:40:97:03:7c  00:30:18:a6:06:eb  /192.168.1.1 : 5431  /192.168.1.4 : 24322
8847  TCP  00:1e:40:97:03:7c  00:30:18:a6:06:eb  /192.168.1.1 : 5431  /192.168.1.4 : 24322
8848  TCP  00:30:18:a6:06:eb  00:1e:40:97:03:7c  /192.168.1.4 : 24322  /192.168.1.1 : 5431
8849  TCP  00:1e:40:97:03:7c  00:30:18:a6:06:eb  /192.168.1.1 : 5431  /192.168.1.4 : 24322
8850  TCP  00:30:18:a6:06:eb  00:1e:40:97:03:7c  /192.168.1.4 : 24322  /192.168.1.1 : 5431
8851  UDP  00:30:18:a6:06:eb  00:1e:40:97:03:7c  /192.168.1.4 : 7048  /202.56.224.153 : 53
8852  UDP  00:1e:40:97:03:7c  00:30:18:a6:06:eb  /202.56.224.153 : 53  /192.168.1.4 : 7048
8853  TCP  00:30:18:a6:06:eb  00:1e:40:97:03:7c  /192.168.1.4 : 24308  /103.5.198.219 : 80
8854  TCP  00:30:18:a6:06:eb  00:1e:40:97:03:7c  /192.168.1.4 : 24325  /103.5.198.219 : 80
8855  UDP  00:30:18:a6:06:eb  00:1e:40:97:03:7c  /192.168.1.4 : 7048  /202.56.224.153 : 53
8856  UDP  00:1e:40:97:03:7c  00:30:18:a6:06:eb  /202.56.224.153 : 53  /192.168.1.4 : 7048
8857  TCP  00:1e:40:97:03:7c  00:30:18:a6:06:eb  /103.5.198.219 : 80  /192.168.1.4 : 24325
8858  TCP  00:30:18:a6:06:eb  00:1e:40:97:03:7c  /192.168.1.4 : 24325  /103.5.198.219 : 80
8859  TCP  00:30:18:a6:06:eb  00:1e:40:97:03:7c  /192.168.1.4 : 24325  /103.5.198.219 : 80
8860  TCP  00:30:18:a6:06:eb  00:1e:40:97:03:7c  /192.168.1.4 : 24326  /220.226.203.110 : 80
8861  TCP  00:1e:40:97:03:7c  00:30:18:a6:06:eb  /103.5.198.219 : 80  /192.168.1.4 : 24325
8862  TCP  00:1e:40:97:03:7c  00:30:18:a6:06:eb  /103.5.198.219 : 80  /192.168.1.4 : 24325
8863  TCP  00:1e:40:97:03:7c  00:30:18:a6:06:eb  /220.226.203.110 : 80  /192.168.1.4 : 24326
8864  TCP  00:30:18:a6:06:eb  00:1e:40:97:03:7c  /192.168.1.4 : 24326  /220.226.203.110 : 80
8865  TCP  00:30:18:a6:06:eb  00:1e:40:97:03:7c  /192.168.1.4 : 24326  /220.226.203.110 : 80
8866  TCP  00:1e:40:97:03:7c  00:30:18:a6:06:eb  /220.226.203.110 : 80  /192.168.1.4 : 24326
```

Figure:- 2 Snap Short Showing captured packets report

The following figure is the main application part of the interface. Which shows the captured packets in a network. The processes of packet capturing discussed in previous section is implemented here and shows the result of application. Which displays the port number , the type of a packet to be captured and this field can be selected according to the use with the help of a filter provided by the interface. The use of this filter is to select the particular type of Packet type which we want to capture. This interface shows the source MAC address , Destination MAC address , source IP address and port and destination IP address and port number. The snap short in figure 4 shows the spoofed packets In the network



Nr.	DATE	LOG
22	Wed Apr 11 10:04:59 GMT+05:30 2012	UNKNOWN MAC DETECTED (C0:3F:0E:8C:A6:02)
24	Wed Apr 11 10:04:59 GMT+05:30 2012	UNKNOWN MAC DETECTED (00:09:0F:C9:9E:A6)
27	Wed Apr 11 10:05:01 GMT+05:30 2012	UNKNOWN MAC DETECTED (44:87:FC:A7:81:65)
29	Wed Apr 11 10:05:43 GMT+05:30 2012	UNKNOWN MAC DETECTED (44:87:FC:A7:7D:DF)
31	Wed Apr 11 10:06:57 GMT+05:30 2012	UNKNOWN MAC DETECTED (44:87:FC:A1:53:2A)
33	Wed Apr 11 10:07:36 GMT+05:30 2012	UNKNOWN MAC DETECTED (01:00:5E:40:00:00)

Figure:- 3 Snap Short showing spoofed packets report

4 CONCLUSION

This application has been used to test the capturing capabilities of API and the result of the monitoring is up to mark as it captures all types of packets demanded by the interface. The article presents a simple way of capturing the network traffic using the functions supplied by the WinPCap Architecture. This application contains a lot of useful primitives and it also provide facilities to capture the spoofed packets and analyze them and to protect our network from intrusions. This paper focuses on the overall load in a particular node the traffic coming to the particular node is reported in figure 3 and also the capturing of spoofed packets that can cause harm to our enterprise is reported in figure 4.

Refrences:-

[1] iac.dtic.mil/iatac/download/intrusion_detection.pdf
[2] sectools.org/tag/sniffers/
[3] www.winpcap.org/
[4] www.carnal0wnage.com/papers/TCPdumpBasics.pdf
[5] en.wikipedia.org/wiki/IEEE_802.3
[6] Rethinking Enterprise Network Control Mart´ın Casado, Member, IEEE, Michael J. Freedman, Member, IEEE, Justin Pettit, Member, IEEE, Jianying Luo, Member, IEEE, Natasha Gude, Member, IEEE, Nick McKeown, Fellow, IEEE, and Scott Shenker, Fellow, IEEE.
[7] A SIMPLE WAY TO CAPTURE NETWORK TRAFFIC: THE WINDOWS PACKET CAPTURE (WINPCAP) ARCHITECTURE Mihai Dorobanţu, M.Sc., Mihai L. Mocanu, Ph.D.
[8] An Architecture for High Performance Network Analysis Fulvio Risso and Loris Degioanni Dipartimento di Automatica e Informatica – Politecnico di Torino
[9] Plab: a packet capture and analysis architecture Alberto Dainotti and Antonio Pescap.
[10] NetCap: A Packet Sniffer in Java 1Rajan Parmar, 2Hetal Patel 1,2Nirma University, Ahmedabad, India.
[11] Detecting Spoofed Packets Steven J. Templeton, Karl E. Levitt Department of Computer Science U.C. Davis
[12] www.winpcap.org/docs/docs_41b5/html/group__packetapi.html
[13] dll.paretologic.com/detail.php/wpcap
[14] netresearch.ics.uci.edu/kfujii/JPCAP/doc/tutorial/index.html
[15] IP Monitoring and Filtering By Gnanambal Chithambaram, Sandeep Dubey Smrithi Barrenkula , Subraja Krishnamurthy , Sucheta P Kodali