*Send Orders for Reprints to reprints@benthamscience.ae*

*The Open Automation and Control Systems Journal*, **2015**, *7*, 167-172 

Open Access

# Design and Development of Network Application Layer Sniffer Analysis Software

Yujiao Wang and Haiyun Lin*

*Department of Physical Science and Technology, Kunming University, Kunming, 65021, China*

**Abstract:** Through analysis of the sniffer technology, a software has been designed for analyzing the network data package of the application layer. It has good user interface and strong adaptability as well as it can help the network administrator to identify and eliminate any network abnormalities and improve the monitoring and guarantee capability of the network's reliability.

**Keywords:** Sniffer, Application Layer, Development of Software, Source Code of Software.

## 1. INTRODUCTION

The network data package analysis software acquires and filters the transmitted network data package on the computer network, verifies and analyzes the information and data of the data package, realizes the recombination and recovering of the data, and then presents a detailed protocol decoder [1]. This can be used to provide different network statistical data, network status information, and error information and obtain the status of the network flow data in order to identify the existing potential security problem in the network. It is an effective tool for the network security management personnel to manage network and analyze the operation status of the network.

## 2. OVERVIEW OF SNIFFER TECHNOLOGY

The sniffer is a passive attack tool with great threat. This tool can monitor the status of the network, data flow and the information transmitted on a network. When the information is transmitted in the form of explicit terms on a network, it can be attached to the network via the process of network monitoring [2]. And once the network interface is set to the monitoring mode, the sniffer can intercept and capture the continuous transmission of information on the network.

## 3. SNIFFER OPERATING PRINCIPLE

The data is transmitted in tiny unit named Frame. The frame is composed of several parts. Different parts perform different functions. The frame is formed through the software named network driver. Then, it is sent to the network cable through the network card, where it reaches its targeted machine and carries out the reverse course on one end of the targeted machine. The Ethernet card of the receiving end of the machine captures the frames, and tells the operating system that the frames have arrived, and then stores them. However during this transmission and reception process the sniffer will bring some security problems. Each workstation on LAN has its own hardware address. These addresses solely represent the machine on the network, which is relatively similar to the internet address system. When a user sends a data package, if it is a broadcast packet, it can arrive at all machines in the LAN. If it is a uni-broadcast packet, it can only reach the machine in the same crash domain. Under a normal situation, all machines on the internet can "hear" the passing flow, but do not respond with their own data package [3]. In other words, Workstation A cannot capture the data of Workstation B, it simply neglects that data. If the network interface of a certain workstation is under the promiscuous mode (the concept of the promiscuous mode is interpreted as the following), it can capture all data packages and frames on the network.

## 4. NETWORK MONITORING PRINCIPLE

The Sniffer procedure is a tool for setting the network interface card (NIC, in general an Ethernet card) into the promiscuous mode through the properties of the Ethernet network. Once the network card is set into this mode, it can accept each information package transmitted on the network. Under the general situation, the network card only can accept the information package related to its own addresses, i.e. the information package transmitted to the local host. To make the sniffer accept and process the information of such approach, the system should support BPF, and support socket – packet under Linux. However, in general, the network hardware and TCP/IP stack do not support receiving or sending the data package unrelated to the local computer. Therefore, to bypass the standard TCP/IP stack, the network card should be set into the promiscuous mode we talked about earlier. In general, to activate this approach, the core must support this pseudo device filter, and also the root authority is required to run this procedure. Therefore, sniffer needs the root to install the identity. If it enters the system with a local user identity,

*Address correspondence to this author at the Department of Physical Science and Technology, Kunming University, Kunming, 65021, China;
Tel: 13987180067; E-mail:tjwwyj817@163.com

1874-4443/15 

it is impossible to detect the code of the root, as then it becomes impossible to run sniffer. Based on the code of sniffer, it is feasible to analyze all information packages and describe the network structure and adopted machines [4]. As it receives any data package transmitted on the same network segment, it can capture passwords, all information passing through the network, confidential files, and other no-cryptographic information.

## 5. INSTALLATION AND DEPLOYMENT OF THE SOFTWARE

The network application layer sniffer analysis software works in the form of sniffing. It collects the original data package in the network, so that it can accurately analyze the network default. If it is installed improperly, there will be great difference in the collected data package, which will influence the analysis result and lead to the above mentioned problems. Therefore, it is quite necessary to properly install and deploy the network protocol analysis software [5].

In general, the installation and deployment of the network protocol analysis software has the following types: share-based internet use Hub as the internet for switching equipment in the center of the network, namely, the share-based network, and the Hub works in the physical layer in OSI layer with the sharing mode. If the central switch equipment of your LAN is Hub, it is feasible to install the networking protocol analysis software on any host in the LAN, as at this time the software can capture all data communication in the whole network. Adopt the switch as the network for the central swift equipment in the network, namely, switch network. The switch works on the data chain layer of the OSI model [6]. Its ports can effectively separate the collision domain. The internet connected by the switch separates the whole internet into many small domains. If the switch of your network has a mirror image function, it is feasible to allocate terminal mirror image on the switch and install the network protocol analysis software on the host that connects the terminal of the mirror image, so the software can capture all data communication from the entire network. Some simple switches do not have the mirror image function, making it impractical to conduct the internet monitoring analysis by port mirroring through these switches. Under such situation, it is feasible to concatenate a Tap or Hub between the switch and the router (or firewall) to finish the data capture. Under a real situation, the topological structure of the network tends to be more complex. During the network analysis we do not need to analyze the whole network; instead, we need to analyze the departments or network segmentation with abnormal operations which makes it feasible to easily realize the data capture of any department or any network segmentation. In a current small network, a large part may still surf on internet by sharing the proxy server. With regard to the analysis of this network the network analysis software can be directly installed on the proxy server. It should be noted that the analysis under such situations require data to be captured on the internal NIC and external NIC of the proxy server.

## 6. REALIZATION OF THE NETWORK APPLICATION LAYER SNIFFER ANALYSIS SOFTWARE

This software can consult the hierarchical structure of the data package, content of the data link layer, content of the network layer, and content and data of the transmission layer. It is feasible to consult and capture the data package through the set filtration, and analyze the content of the data package. The software, developed through VS2005 is a necessary software tool for the network maintenance, with simple interface and convenient utilization.

## 7. SOME SOURCE CODE REALIZED BY THE SOFTWARE

Protocol, port, source, target address, package length, information length, and other source codes

```
public class IPPacketMessage : EventArgs
    {
        private static Filter myfilter = Filter.Get_instance();
        private string protocol;
        private string destination_port;
        private string origination_port;
        private string destination_address;
        private string origination_address;
        private string ip_version;
        private unit total_packet_length;
        private unit message_length;
        private unit header_length;
        private byte[] receive_buf_bytes = null;
        private byte[] ip_header_bytes = null;
        private byte[] message_bytes = null;
        public IPPacketMessage()
        {
            this.protocol = "";
            this.destination_port = "";
            this.origination_port = "";
            this.destination_address = "";
            this.origination_address = "";
            this.ip_version = "";

            this.total_packet_length = 0;
            this.message_length = 0;
```

```
        this.header_length = 0;
      this.receive_buf_bytes = new byte[IPPacketIfc.rcv_buf_len];
      this.ip_header_bytes = new byte[IPPacketIfc.rcv_buf_len];
      this.message_bytes = new byte[IPPacketIfc.rcv_buf_len];
    }
    public string Protocol
    {
        get { return protocol; }
        set { protocol = value; }
    }
    public string Destination Port
    {
        get { return destination_port; }
        set { destination_port = value; }
    }
    public string OriginationPort
    {
        get { return origination_port; }
        set { origination_port = value; }
    }
    public string DestinationAddress
    {
        get { return destination_address; }
        set { destination_address = value; }
    }
    public string OriginationAddress
    {
        get { return origination_address; }
        set { origination_address = value; }
    }
    public string IPVersion
    {
        get { return ip_version; }
        set { ip_version = value; }
    public unit PacketLength
    }
    {
        get { return total_packet_length; }
        set { total_packet_length = value; }
    }
    public unit MessageLength
    {
        get { return message_length; }
        set { message_length = value; }
    }
    public unit HeaderLength
    {
        get { return header_length; }
        set { header_length = value; }
    }
    public byte[] ReceiveBuffer
    {
        get { return receive_buf_bytes; }
        set { receive_buf_bytes = value; }
    }
    public byte[] IPHeaderBuffer
    {
        get { return ip_header_bytes; }
        set { ip_header_bytes = value; }
    }
    public byte[] MessageBuffer
    {
        get { return message_bytes; }
        set { message_bytes = value; }
    }
public uint HeaderLength
    {
        get { return header_length; }
        set { header_length = value; }
    }
    public byte[] ReceiveBuffer
    {
```

```
    get { return receive_buf_bytes; }
    set { receive_buf_bytes = value; }
}
public byte[] IPHeaderBuffer
{
    get { return ip_header_bytes; }
    set { ip_header_bytes = value; }
}
public byte[] MessageBuffer
{
    get { return message_bytes; }
    set { message_bytes = value; }
}
unsafe public void Translate(byte[] message,int len)
{
    byte temp_protocol=0;
     unit temp_version=0;
     unit temp_ip_srcaddr=0;
     unit temp_ip_destaddr=0;
     short temp_srcport=0;
     short temp_dstport=0;
     IPAddress temp_ip;
    fixed (byte* fixed_buf = message)
    {
      IPHeader* head = (IPHeader*)fixed_buf;
      this.HeaderLength = (uint)(head->ip_verlen & 0x0F) << 2;
      temp_protocol = head->ip_protocol;
      temp_version = (unit)(head->ip_verlen & 0xF0) >> 4;
      this.IPVersion = temp_version.ToString();
      temp_ip_srcaddr = head->ip_srcaddr;
      temp_ip_destaddr = head->ip_destaddr;
      temp_ip = new IPAddress(temp_ip_srcaddr);
      this.OriginationAddress = temp_ip.ToString();
      temp_ip = new IPAddress(temp_ip_destaddr);
      this.DestinationAddress = temp_ip.ToString();
      temp_srcport = *(short*)&fixed_buf[this.HeaderLength];  //for the purpose of getting two byte
      temp_dstport = *(short*)&fixed_buf[this.HeaderLength + 2];  //for the purpose of getting two byte
      this.OriginationPort = ((ushort)IPAddress.NetworkToHostOrder(temp_srcport)).ToString();
      this.DestinationPort = ((ushort)IPAddress.NetworkToHostOrder(temp_dstport)).ToString();
      this.PacketLength = (unit)len;
      this.MessageLength = (unit)len - this.HeaderLength;
      this.ReceiveBuffer = message;
      Array.Copy(message, 0, this.IPHeaderBuffer, 0, (int)this.HeaderLength);
      Array.Copy(message, (int)this.HeaderLength, this.MessageBuffer, 0, (int)this.MessageLength);
      switch (temp_protocol)
      {
        case 1: this.Protocol = "ICMP"; break;
        case 2: this.Protocol = "IGMP"; break;
        case 6: this.Protocol = "TCP"; break;
        case 17: this.Protocol = "UDP";   break;
        default: this.Protocol = "UNKNOWN"; break;
      }
    }
}
```

## 8. MAIN INTERFACE DESIGN

The main interface design after the operation of the software is shown as follows:

Input IP in IP address, click “start” button, and start the sniffing operation. Click “suspend” to temporarily stop the sniffing operation, and click “start” to continue the operation from the chasm. Click “stop” to completely stop this sniffing operation. Click “eliminate” to eliminate the content displayed by this operation (Fig. **1**).

## 9. STATEMENT INTERFACE

Some listed information, and the detailed statement is presented below which shows different statement content of

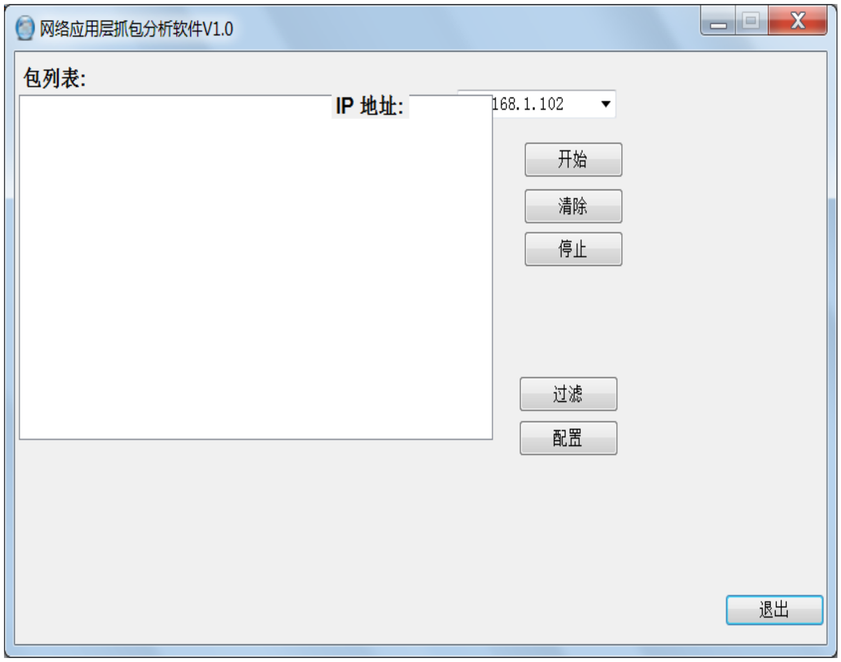


**Fig. (1).** The main interface design.

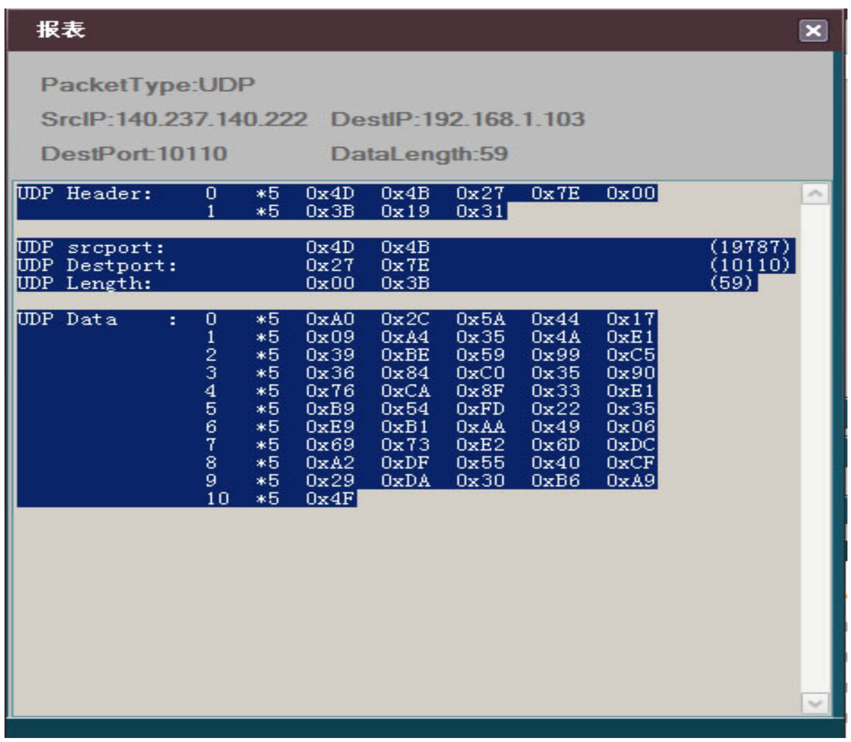


**Fig. (2).** Content statement of UDP.

UDP, TCP and ICMP. The length of the header is 4 byte. The purpose of this field is to describe the length of IP package head, as there is only a selective part that is lengthened. The minimum length of IP package head is 20 byte. The maximum length of selective part with the extending length may be turned into 24 byte, SrcPort: identify the point of the superstratum source processor to receive TCP service; DestPort: identify the point of the superstratum source processor to receive TCP service; Length: 16 byte; The maximum length of IP package is 64,535 byte; Data: including the superstratum information (Fig. **2**).

Content statement of TCP is largely identical but with minor difference with that of UDP. The former one has additional Data Offset. Data Offset: 4 byte. 32-byte serial number in TCP protocol presents the starting position of the data (Fig. **3**).

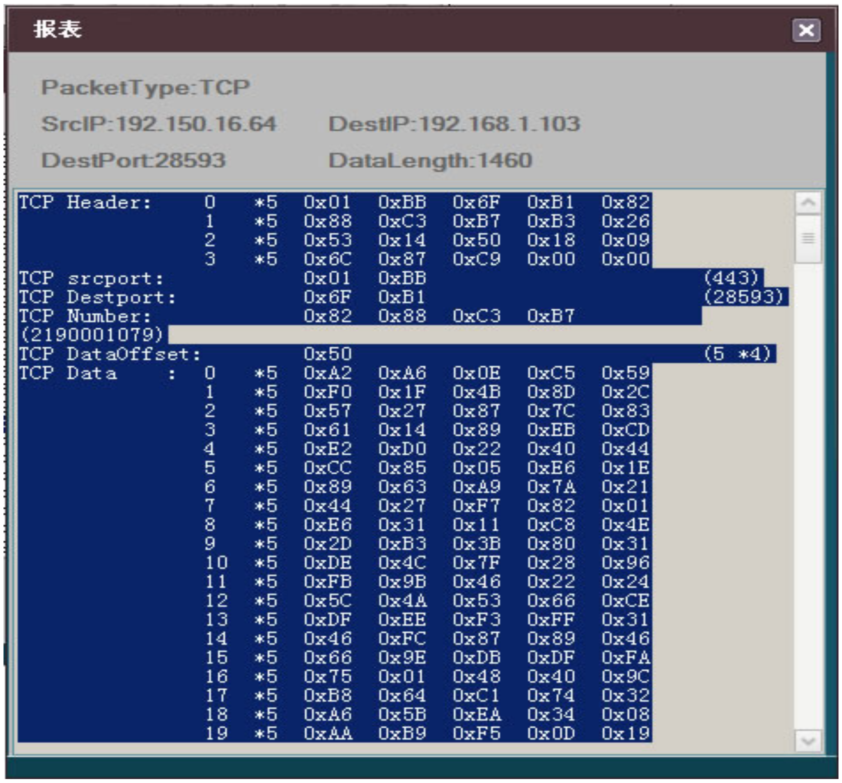


**Fig. (3).** Content statement of TCP.

## CONCLUSION

The software testing result indicates that the software has fast speed in capturing data, and it can consult the hierarchical structure of the data package, content of the data link layer, content of the network layer, and content and data of the transmission layer; it is feasible to consult and capture the data package through setting the filtration, and analyze the content of the data package. The analysis result of this software is correct, and the content is with distinct gradation and easy to understand, which powerfully supports the work of the network maintenance.

## CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.


## ACKNOWLEDGEMENTS

Declared none.


## REFERENCES


[1] U. Lamping, and R. Sharpe, *Wireshark User's Guide* [E-book], 2010.

[2] How to capture the packet. www.wireshark.org. 2011 [Online].

[3] Capturing packets on windows. http://www.wireshark.org/faq.html-#sec8.html.2011[Online].

[4] D. Liu. "The technology research based on network packet," *J. Jilin Univ.*, 2012.

[5] F. Liu, and F. Yang, "Capture the network system based on WinPcap," *Comput. CD Software Appl.*, vol. 8, pp. 132-133, 2012.

[6] S. Du, J. Wang, Z. Chen, and L. Xiang, "Research on the testing techniques of the software network interface based on the capture and analysis of the packet," *Sci. Res.*, vol. 8, pp. 1305-1308, 2011.



Received: December 15, 2014 Revised: January 04, 2015 Accepted: February 25, 2015