An Application System to Detect Network Intruder

L.S. Yu,
School of Information Science and Engineering
Central South University
Changsha, Hunan, P.R. China

K. Chanthala
School of Information Science and Engineering
Central South University
Changsha, Hunan, P.R. China

Abstract--Packet Monitoring is an essential part of network security. In this paper, an application system is implemented to detect intruders by monitoring the packets over the network. It will alert the end users once the application detects the intruder. This application system also gives the ability to see the details of the packets which is being captured. After analyzing the packets the user can take the necessary action to prevent the intruder from gaining access to the user's machine and to prevent the intruders from attacking the user's machine. The experiment results show that the approach is highly effective and stable.

Keywords-packet monitoring; network security; intrusion detection

I. Introduction

With globalization the global economy driven by e-business has extended so vast that attacks on the networks are becoming more frequent, more various and more costly[1]. Thus providing protection for information assets has driven companies in search for privacy to be protected. The different phases in network intrusion are 'Before', 'During' and 'After' the attack[2]. It is very easy to break it down this way, but very hard to combat how someone might gain access to your network. In fact, research from the FBI and other security research groups contend that about 50%-75% of all intrusions have taken place from inside your corporate "trusted" network[3]. Additionally, if intrusions are performed for the purpose of gathering information or resource usage, rather than denial of service or defamatory purposes, the attacker will do everything in his or her power station to ensure that their presence is difficult to detect. This makes it even harder to detect and trace what is happening within your network environment.

Sniffing Technologies can help to respond to attacks in progress. When deployed within a network, a sniffer[1-4] can help to alert administrators of intrusions, filter traffic based on specific criteria, and capture data/evidence to help reconstruct a record of what systems have actually been affected or compromised which can be used for analysis at a later phase. Sniffer logs can provide invaluable data for forensic analysis. In this paper how a sniffer system can be used to complement and enhance traditional security monitoring techniques has been explained.

II. Requirements for the Application System

In this paper the .NET framework[5] has been used to design the application system for packet sniffing in the System to Detect Network Intruder. The application system is capable of effectively detect intrusion on a network and alerts the network administrator on any suspicious network activity that is being done on the network. Pre-determination of valid network users are required for the detection. The software uses Red algorithm to identify users which are possibly flooding the network and acts as malicious users.

There are several key functions[6-7] that are required in the design of the system to detect network intruder.

• Selecting Network Interface Card (NIC) for the monitoring of packets sent on that network.

• Allowing details of network packet header information to be viewed by the user upon demand.

• Unauthorized users will be highlighted in red, and alerts will be given to the administrator for such intruding users.

• Authorized users can be added to the intruder list on demand by the network administrator (user of this program).

• Network Administrator can also specify bounds of quantity of packets and length of time for any period to monitor the network for flood attacks. IP addresses exceeding the bounds supplied will be shown in a dialog box as "Marked Packets". These IP addresses can then be subsequently added to the IP Intruder List.

• Creating a log file containing all Intruding IP Addresses for later network analysis by the Network Administrator.

• Network Administrator may also remove IP Addresses from the Intruder List if he feels they are not intruders.

The intruder detection system shall accurately meet the demands of the customer and the outlined strategy should be highly efficient. The system may be easily expanded to include minor functional enhancements.

III. Implementation for the Application System

A. *Modules for the Application System*

Three modules are required for this system:

Packet Monitoring: This will facilitate the administrator to listen for Incoming & Outgoing packets of data of a particular system connected to the network and get a clear knowledge of the underlying traffic along the network[6-8].

Packet Information Decoding: The administrator can concern with a detailed list of all the specifications of packets being transmitted along the Network. This information will include parameters such as IP version number, time to live, source Address and port number, destination address and port number and other information contained in IP and TCP headers.

Red Algorithm Implementation[8]: In this module users are able to monitor network traffic and view a summary of IP Addresses to check for the particular flood attack in question.

B. Packet Capturing

The Main Application contains a constructor method which initializes the form and displays the designed interface. Since it will attempt to monitor packets being transmitted over a specified Network Interface Card, the user of the application must first have the appropriate privileges. The following Code excerpt ensures that the current user is a System Administrator if it returns 0 values:

int ret = AllocateAndInitializeSid(NtAuthority, 2, SECURITY_BUILTIN_DOMAIN_RID, DOMAIN_ALIAS_RID_ADMINS, 0, 0, 0, 0, 0, 0, out AdministratorsGroup);

CheckTokenMembership(IntPtr.Zero, AdministratorsGroup, ref ret)

Upon the user has the minimum access rights to effectively execute the designed program, a thread is created to start monitoring the raw data stream passing through the specified interface card. This is done using the socketObject created on the IP protocol to receive Raw Data.

m_Monitor = new Socket(AddressFamily.InterNetwork, SocketType.Raw , ProtocolType.IP);

m_Monitor.IOControl(SIO_RCVALL, BitConverter.GetBytes((int)1), null);

Where m_Monitor is used to capture all IP Packets. SIO_RCVALL is a control code enables a socket to receive all IPv4 or IPv6 packets passing through a network interface. Once the network packet has been received, AsyncCallback event will trigger a function called OnReceive.

int received = m_Monitor.OnReceive(ar)

where received contains information about number of bytes received from the network interface and then copy the network packet into byte array for later processing.

Array.Copy(Buffer, 0, packet, 0, received);

Buffer content will be stored into the packet from the 0th index. Once the contents are copied into packet byte array, an instance of the Packet parsing module is created by sending the captured packet content.

OnNewPacket(new Packet(packet));// creates instance of Packet class for Parsing.

C. Packet Parsing Module

Prior to the actual Parsing of the received packet, it is crucial to enumerate various core variables and their possible values to facilitate the ease of comparisons. If the precedence designations are of concern to a particular network, it is necessary to control the access to, and using of, those precedence designations.

```
        public enum Precedence {
                Routine = 0,
                Priority = 1,
                Immediate = 2,
                Flash = 3,
        FlashOverride = 4,
        CRITICECP = 5,
        InternetworkControl = 6,
        NetworkControl = 7
        }
```

The use of the Delay, Throughput, and Reliability indications may increase the cost. In many situations they are compete forces.

```
        public enum Delay {
          NormalDelay = 0,
          LowDelay = 1
        }
        public enum Throughput {
          NormalThroughput = 0,
          HighThroughput = 1
         }
        public enum Reliability {
          NormalReliability = 0,
          HighReliability = 1
        }
```

The following fields indicate the protocol used in the data portion of the internet datagram.

```
        public enum Protocol {
                Ggp = 3,
                Icmp = 1,
                Idp = 22,
                Igmp = 2,
                IP = 4,
                ND = 77,
                Pup = 12,
                Tcp = 6,
                Udp = 17,
                Other = -1
        }
```

The followings are used to sniff and decode the packet, Raw[0] holds information about the first byte of the network,version is stored in the higher order byte of Raw[0]

m_Version = (raw[0] & 0xF0) >> 4;

After the operation , Lower byte will be masked and exact value stored in the version information of the header can be got.

m_HeaderLength = (raw[0] & 0x0F) * 4 /*sizeof(int)

Similarly the following code can extract other information :

```
m_Precedence = (Precedence)((raw[1] & 0xE0) >> 5);
m_Delay = (Delay)((raw[1] & 0x10) >> 4);
m_Throughput = (Throughput)((raw[1] & 0x8) >> 3);
m_Reliability = (Reliability)((raw[1] & 0x4) >> 2);
m_TotalLength = raw[2] * 256 + raw[3];
m_Identification = raw[4] * 256 + raw[5];
m_TimeToLive = raw[8];
if (Enum.IsDefined(typeof(Protocol), (int)raw[9]))
  m_Protocol = (Protocol)raw[9];
else
```

```
    m_Protocol = Protocol.Other;
m_Checksum = new byte[2];
m_Checksum[0] = raw[11];
m_Checksum[1] = raw[10];
```

The information will be stored in the class data member, which will be used to identify intruder. The following code can be used to parse source and destination IP.

```
m_SourceAddress = new –
IPAddress(BitConverter.ToUInt32(raw, 12));
m_DestinationAddress = new –
IPAddress(BitConverter.ToUInt32(raw, 16));
```

IV. EXPERIMENTS AND RESULTS

We do the experiment in our lab. Once the application starts, it will display all the current network interfaces cards that are connected to the network with their current IP addresses .Upon selection of an IP address to start monitoring, all the network traffic passing through that interfaces will be monitored and captured, and a summary is presented to the user including Time, Protocol used, Source, Destination and Length of the packet (Shown as Figure Ⅰ).

The Administrator may be informed of additional details of any packet (Shown as Figure Ⅱ) by simply double clicking on any item. It includes the IP version, Port numbers and the data contained in the packet. It may be particularly important for an experienced and highly trained network administrator to get a clear picture of the kind of transactions and then to detect the intruders.

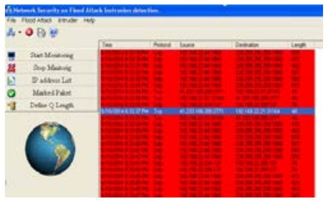

FIGURE I. MONITORING INTERFACE

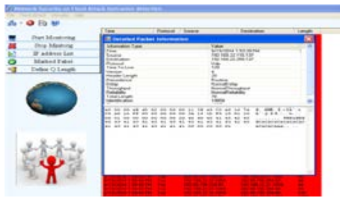

FIGURE II. DETAILED PACKET INFORMATION

The Log file is saved in the default directory at C:\Intruders.txt which contains the log summary of all Intruding IP’s, with the specification of the Time of the Attack, the Protocol used, and the Source and Destination Addresses. When the user stipulates the details of the flood attack, he is prompted to define the Queue Length, which is the maximum number of packets being transmitted across the network. The user must also define a Time (in seconds) for the Red Algorithm to monitor for the flood. Upon the completion of the flood attack monitoring, users may view all of the IP addresses that have transmitted information above the predefined packet amount in the specified time by clicking on the Flood Attack Menu Item. The Marked Packet displays a summary of the detected IP address that has exceeded the packet count value in the allowed time specified by the user.

The administrator can add any IP that has been identified by the Red Algorithm as flooding the network, to the Intruder IP List which contains all blacklisted IP’s and therefore be highlighted in red, and then give a sound alarm to prompt the administrator.

The BlackList IP holds all the IP Addresses and corresponding port numbers that have been detected as intruders either by one of the following ways: process of discovery and non-authentication using the Hello Protocol. Upon click the Marked Packet List, the result of the execution summary will be displayed using the Red Algorithm. IP addresses can also be manually added to the Intruder List by accessing the Intruder Menu.

V. CONCLUSION

The application system to detect network intruder is an administrative tool which empowers the administrator to take the necessary action if malicious packets are observed. The tool can also automatically alert the administrator in critical situations such as illegal intrusion, repeated logins and dropping certain specified packets. It can also keep the user safe by monitoring the packets over the network and prevent any malicious packets to harm the user. The experiment results shows its effective and high efficient for the security and protection the user from malicious activities.

REFERENCES

[1] Abdul Nasir Khan, Kalim Qureshi, and Sumair Khan, An Intelligent Approach of Sniffer Detection , The International Arab Journal of Information Technology, Vol. 9, No. 1, January 2012, pp.9-15

[2] Packet Sniffing with Wireshark and Tcpdump, http://science.hamptonu.edu/compsci/docs/iac/packet_sniffing.pdf

[3] Yuan X., Vega P., Xu J., Yu H., and Li Y.,Using Packet Sniffer in Class Experience and Evaluation, in Proceedingss of the 45 th Annual Southeast Regional Conference ACMSE , USA,pp. 116-121, 2007.

[4] Sniffer Technologies – A Network Associates Business , http://www.packetattack.com/pdf/whitepaper.pdf

[5] Andrew Troelsen: C# and the .NET platform, Springer Private limited, New Delhi, Second edition, 2008

[6] Pallavi Asrodia,Hemlata Patel . Network Traffic Analysis Using Packet Sniffer,International Journal of Engineering Research and Applications (IJERA) Vol. 2, Issue3,May-Jun2012, pp.854-856

[7] Qadeer M.A., Zahid M., Iqbal A., Siddiqui M.R. Network Traffic Analysis and Intrusion Detection Using Packet Sniffer. ICCSN‘10 Second International Conference, 2010, pp: 313-317

[8] Bo Yu. Based on the network sniffer implement network monitoring. 2010 International Conference on Computer Application and System Modeling (ICCASM), Volume: 7 , 2010, pp: V7-1-V7-3