

CHƯƠNG 13 PHÂN TÍCH GÓI TIN MẠNG

ThS. Trần Bá Nhiệm
Website:
sites.google.com/site/tranbanhiem
Email: tranbanhiem@gmail.com

Nội dung

- Giới thiệu
- Phân tích mức Network (IP-level)
- Phân tích mức Data-link
- Phân tích mức Physical

Giới thiệu

- Trong các chương trước chúng ta đã làm việc có liên quan đến việc di chuyển dữ liệu từ client đến server, nhưng khi chúng ta chưa khảo sát kỹ cái gì đang di chuyển giữa chúng
- Hầu như chúng ta không cần quan tâm dữ liệu gì được nhận bởi các ứng dụng.

6/29/2011

Chương 13: Phân tích gói tin mạng

3

Giới thiệu

- Bắt gói tin không phải là vấn đề mới, nhưng có một số ứng dụng thực sự dùng công nghệ này để viết virus hoặc do thám máy tính
- Phần mềm xử lý ở mức gói tin có thể ứng dụng được trong thương mại, ví dụ phát hiện ứng dụng nào đang lạm dụng băng thông để cảnh báo,...

6/29/2011

Chương 13: Phân tích gói tin mạng

4

Giới thiệu

- Phần mềm kiểm tra lưu thông cũng có thể được dùng để phát hiện các gói tin của virus, việc dùng phần mềm lậu, giả mạo email, tấn công mạng,...
- Tấn công mạng kiểu DoS có thể phát hiện được bởi một số lượng lớn gói tin “xấu” gửi liên tục vào server

6/29/2011

Chương 13: Phân tích gói tin mạng

5

Giới thiệu

- Tấn công kiểu ping-of-death là một số lượng lớn kết nối TCP không hoàn tất có thể phát hiện nhờ hiện tượng SYN flood mà server nạn nhân không biết vẫn cố thử gửi ACK cho kẻ tấn công và không hề nhận được phản hồi
- Việc dùng phần mềm lậu giúp phát hiện nhân viên tiêu tốn thời gian chơi game,...

6/29/2011

Chương 13: Phân tích gói tin mạng

6

Giới thiệu

- Kiểm tra lưu thông email có thể giúp phát hiện nhân viên tiết lộ bí mật cho đối thủ, ngăn chặn giả mạo email.
- Ứng dụng có thể quản lý IP máy tương ứng với địa chỉ email, trong trường hợp không so trùng thì có thể hiển thị cảnh báo cho người dùng hoặc người quản trị hệ thống để có biện pháp xử lý

6/29/2011

Chương 13: Phân tích gói tin mạng

7

Phân tích mức IP

- Phân tích ở mức này liên quan đến TCP/IP và UDP, cũng như các dịch vụ chạy trên nó như DNS, HTTP, FTP, ...
- Tại mức này, chúng ta không cần phần mềm nào đặc biệt, bởi vì mọi thứ đã được .NET hỗ trợ

6/29/2011

Chương 13: Phân tích gói tin mạng

8

Hiện thực phân tích mức IP

- Tạo project mới, gồm 1 form, 1 Listbox tên lbPackets, 2 button tên btnStart, btnStop
- Khai báo biến public:
public Thread Listener;
- Xử lý sự kiện Click của button btnStart:
private void btnStart_Click(object sender, EventArgs e)
{
 btnStart.Enabled = false;
 btnStop.Enabled = true;
 Listener = new Thread(new ThreadStart(Run));
 Listener.Start();
}

6/29/2011

Chương 13: Phân tích gói tin mạng

9

Hiện thực phân tích mức IP

- Xử lý sự kiện Click của button btnStop:
private void btnStop_Click(object sender, EventArgs e)
{
 btnStart.Enabled = true;
 btnStop.Enabled = false;
 if (Listener != null)
 {
 Listener.Abort();
 Listener.Join();
 Listener = null;
 }
}

6/29/2011

Chương 13: Phân tích gói tin mạng

10

Hiện thực phân tích mức IP

- Phần quan trọng nhất của chương trình này chính là hàm Run().
- Hàm Run thực hiện khởi tạo các biến tương thích để lưu trữ dữ liệu gửi/nhận, thiết lập socket để kết nối, khi đã thiết lập được thì thực hiện nhận dữ liệu cho đến khi nào kết nối đóng
- Hiện thực hàm Run như sau:

6/29/2011

Chương 13: Phân tích gói tin mạng

11

Hiện thực phân tích mức IP

```
public void Run()
{
    int len_receive_buf = 4096;
    int len_send_buf = 4096;
    byte[] receive_buf = new byte[len_receive_buf];
    byte[] send_buf = new byte[len_send_buf];
    int cout_receive_bytes;
    Socket socket = new
    Socket(AddressFamily.InterNetwork,
    SocketType.Raw, ProtocolType.IP);
```

6/29/2011

Chương 13: Phân tích gói tin mạng

12

Hiện thực phân tích mức IP

```

socket.Blocking = false;
IPHostEntry IPHost =
    Dns.GetHostByName(Dns.GetHostName());
socket.Bind(new
    IPEndPoint(IPAddress.Parse(IPHost.AddressList[0].
        ToString()), 0));
socket.SetSocketOption(SocketOptionLevel.IP,
    SocketOptionName.HeaderIncluded, 1);
byte[] IN = new byte[4] { 1, 0, 0, 0 };
byte[] OUT = new byte[4];

```

6/29/2011

Chương 13: Phân tích gói tin mạng

13

Hiện thực phân tích mức IP

```

int SIO_RCVALL = unchecked((int)0x98000001);
int ret_code = socket.IOControl(SIO_RCVALL, IN,
    OUT);
while (true){
    IAsyncResult ar =
        socket.BeginReceive(receive_buf, 0,
            len_receive_buf, SocketFlags.None, null, this);
    cout_receive_bytes = socket.EndReceive(ar);
    Receive(receive_buf, cout_receive_bytes);
}
}

```

6/29/2011

Chương 13: Phân tích gói tin mạng

14

Hiện thực phân tích mức IP

- Hiện thực hàm Receive:


```
public void Receive(byte[] buf, int len)
{
    if (buf[9] == 6)
    {
        lbPackets.Items.Add(Encoding.ASCII.GetString(buf)
        .Replace("\0", " "));
    }
}
```
- TCP packet luôn có byte thứ 9 trong header có giá trị bằng 6

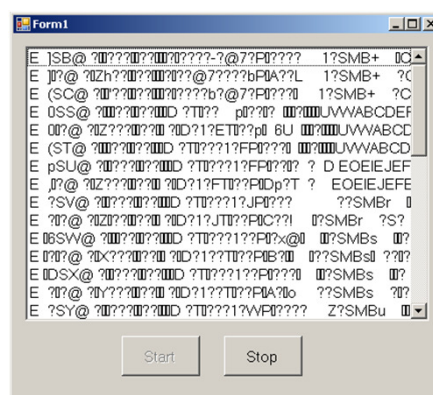
6/29/2011

Chương 13: Phân tích gói tin mạng

15

Hiện thực phân tích mức IP

Chạy ứng dụng, mở một trang web nào đó trên trình duyệt, kết quả tương tự như hình



6/29/2011

Chương 13: Phân tích gói tin mạng

16

Hiện thực Sniffer mức IP

- Tạo project mới, gồm 1 form, 1 Tree View tên treeView, 1 button tên btnStart, 1 combobox tên cmbInterfaces
- Thêm kiểu liệt kê:

```
public enum Protocol
{
    TCP = 6,
    UDP = 17,
    Unknown = -1
};
```

6/29/2011

Chương 13: Phân tích gói tin mạng

17

Hiện thực Sniffer mức IP

- Khai báo 3 biến cấp form:

```
private Socket mainSocket; private byte[]
byteData = new byte[4096];
private bool bContinueCapturing = false;
```
- Trong đó mảng byteData sẽ lưu trữ các dữ liệu đến trên socket và bContinueCapturing là cờ cho biết có bắt được gói tin hay không

6/29/2011

Chương 13: Phân tích gói tin mạng

18

Hiện thực Sniffer mức IP

- Khai báo phương thức delegate cho class:
private delegate void AddTreeNode(TreeNode node);
- Hiện thực hàm AddTreeNode:
private void OnAddTreeNode(TreeNode node)
{
 treeView.Nodes.Add(node);
}
- Xử lý cho sự kiện Load của form:

6/29/2011

Chương 13: Phân tích gói tin mạng

19

Hiện thực Sniffer mức IP

```
private void SnifferForm_Load(object sender, EventArgs e)
{
    string strIP = null;
    IPHostEntry HosyEntry =
    Dns.GetHostEntry((Dns.GetHostName()));
    if (HosyEntry.AddressList.Length > 0) {
        foreach (IPAddress ip in HosyEntry.AddressList) {
            strIP = ip.ToString();
            cmbInterfaces.Items.Add(strIP);
        }
    }
}
```

6/29/2011

Chương 13: Phân tích gói tin mạng

20

Hiện thực Sniffer mức IP

- Xử lý cho sự kiện Closing của form:

```
private void SnifferForm_FormClosing(object sender, FormClosingEventArgs e)
{
    if (bContinueCapturing)
    {
        mainSocket.Close();
    }
}
```

6/29/2011

Chương 13: Phân tích gói tin mạng

21

Hiện thực Sniffer mức IP

- Xử lý cho sự kiện Click của button btnStart:

```
private void btnStart_Click(object sender, EventArgs e)
{
    if (cmbInterfaces.Text == ""){
        MessageBox.Show("Select an Interface to capture the packets.", "MJsniiffer",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    try {
        if (!bContinueCapturing) {
            btnStart.Text = "&Stop";
            bContinueCapturing = true;
        }
    }
}
```

6/29/2011

Chương 13: Phân tích gói tin mạng

22

Hiện thực Sniffer mức IP

```

        mainSocket = new Socket(AddressFamily.InterNetwork,
                                SocketType.Raw, ProtocolType.IP);
        mainSocket.Bind(new
        IPEndPoint(IPAddress.Parse(cmbInterfaces.Text), 0));
        mainSocket.SetSocketOption(SocketOptionLevel.IP,
        SocketOptionName.HeaderIncluded, true);
        byte[] byTrue = new byte[4] {1, 0, 0, 0};
        byte[] byOut = new byte[4]{1, 0, 0, 0};
        mainSocket.IOControl(IOControlCode.ReceiveAll,
        byTrue, byOut);
        mainSocket.BeginReceive(byteData, 0, byteData.Length,
        SocketFlags.None,
        new AsyncCallback(OnReceive), null);
    }

```

6/29/2011

Chương 13: Phân tích gói tin mạng

23

Hiện thực Sniffer mức IP

```

        else {
            btnStart.Text = "&Start";
            bContinueCapturing = false;
            mainSocket.Close ();
        }
    }
    catch (Exception ex) {
        MessageBox.Show(ex.Message, "Sniffer",
        MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}

```

6/29/2011

Chương 13: Phân tích gói tin mạng

24

Hiện thực Sniffer mức IP

```
private void OnReceive(IAsyncResult ar)
{
    try
    {
        int nReceived = mainSocket.EndReceive(ar);
        ParseData (byteData, nReceived);
        if (bContinueCapturing)
        {
            byteData = new byte[4096];
            mainSocket.BeginReceive(byteData, 0,
            byteData.Length, SocketFlags.None,
            new AsyncCallback(OnReceive), null);
        }
    }
}
```

6/29/2011

Chương 13: Phân tích gói tin mạng

25

Hiện thực Sniffer mức IP

```
    }
    catch (ObjectDisposedException)
    {
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Sniffer",
        MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}
```

6/29/2011

Chương 13: Phân tích gói tin mạng

26

Hiện thực Sniffer mức IP

```
private void ParseData(byte[] byteData, int nReceived)
{
    TreeNode rootNode = new TreeNode();
    IPHeader ipHeader = new IPHeader(byteData,
nReceived);
    TreeNode ipNode = MakeIPTreeNode(ipHeader);
    rootNode.Nodes.Add(ipNode);
    switch (ipHeader.ProtocolType)
    {
        case Protocol.TCP:
            TCPHeader tcpHeader = new
TCPHeader(ipHeader.Data,ipHeader.MessageLength);
```

6/29/2011

Chương 13: Phân tích gói tin mạng

27

Hiện thực Sniffer mức IP

```
TreeNode tcpNode =
MakeTCPTreeNode(tcpHeader);
    rootNode.Nodes.Add(tcpNode);
    if (tcpHeader.DestinationPort == "53" ||
tcpHeader.SourcePort == "53")
    {
        TreeNode dnsNode =
MakeDNSTreeNode(tcpHeader.Data,
(int)tcpHeader.MessageLength);
        rootNode.Nodes.Add(dnsNode);
    }
    break;
```

6/29/2011

Chương 13: Phân tích gói tin mạng

28

Hiện thực Sniffer mức IP

```

        case Protocol.UDP:
            UDPHeader udpHeader = new
            UDPHeader(ipHeader.Data,
            (int)ipHeader.MessageLength);
            TreeNode udpNode =
            MakeUDPTreeNode(udpHeader);
            rootNode.Nodes.Add(udpNode);
            if (udpHeader.DestinationPort == "53" ||
            udpHeader.SourcePort == "53") {
                TreeNode dnsNode =
                MakeDNSTreeNode(udpHeader.Data,
                Convert.ToInt32(udpHeader.Length) - 8);

```

6/29/2011

Chương 13: Phân tích gói tin mạng

29

Hiện thực Sniffer mức IP

```

            rootNode.Nodes.Add(dnsNode);
        }
        break;
        case Protocol.Unknown:
            break;
    }
    AddTreeNode addTreeNode = new
    AddTreeNode(OnAddTreeNode);
    rootNode.Text = ipHeader.SourceAddress.ToString() + "-"
+ ipHeader.DestinationAddress.ToString();
    treeView.Invoke(addTreeNode, new object[] {rootNode});
}

```

6/29/2011

Chương 13: Phân tích gói tin mạng

30

Hiện thực Sniffer mức IP

- Một số hàm hỗ trợ cho phần giao diện:

```
private TreeNode MakeIPTreeNode(IPHeader ipHeader)
{
    TreeNode ipNode = new TreeNode();
    ipNode.Text = "IP";
    ipNode.Nodes.Add ("Ver: " + ipHeader.Version);
    ipNode.Nodes.Add ("Header Length: " +
ipHeader.HeaderLength);
    ipNode.Nodes.Add ("Differentiated Services: " +
ipHeader.DifferentiatedServices);
    ipNode.Nodes.Add("Total Length: " +
ipHeader.TotalLength);
```

6/29/2011

Chương 13: Phân tích gói tin mạng

31

Hiện thực Sniffer mức IP

```
ipNode.Nodes.Add("Identification: " +
ipHeader.Identification);
ipNode.Nodes.Add("Flags: " + ipHeader.Flags);
ipNode.Nodes.Add("Fragmentation Offset: " +
ipHeader.FragmentationOffset);
ipNode.Nodes.Add("Time to live: " + ipHeader.TTL);
switch (ipHeader.ProtocolType)
{
    case Protocol.TCP:
        ipNode.Nodes.Add ("Protocol: " + "TCP");
        break;
    case Protocol.UDP:
```

6/29/2011

Chương 13: Phân tích gói tin mạng

32

Hiện thực Sniffer mức IP

```

ipNode.Nodes.Add ("Protocol: " + "UDP");
break;
case Protocol.Unknown:
ipNode.Nodes.Add ("Protocol: " + "Unknown");
break;
}
ipNode.Nodes.Add("Checksum: " + ipHeader.Checksum);
ipNode.Nodes.Add("Source: " +
ipHeader.SourceAddress.ToString());
ipNode.Nodes.Add("Destination: " +
ipHeader.DestinationAddress.ToString());
return ipNode;
}

```

6/29/2011

Chương 13: Phân tích gói tin mạng

33

Hiện thực Sniffer mức IP

```

private TreeNode MakeTCPTreeNode(TCPHeader tcpHeader)
{
    TreeNode tcpNode = new TreeNode();
    tcpNode.Text = "TCP";
    tcpNode.Nodes.Add("Source Port: " + tcpHeader.SourcePort);
    tcpNode.Nodes.Add("Destination Port: " +
tcpHeader.DestinationPort);
    tcpNode.Nodes.Add("Sequence Number: " +
tcpHeader.SequenceNumber);
    if (tcpHeader.AcknowledgementNumber != "")
        tcpNode.Nodes.Add("Acknowledgement Number: " +
tcpHeader.AcknowledgementNumber);
}

```

6/29/2011

Chương 13: Phân tích gói tin mạng

34

Hiện thực Sniffer mức IP

```

tcpNode.Nodes.Add("Header Length: " +
tcpHeader.HeaderLength);
tcpNode.Nodes.Add("Flags: " + tcpHeader.Flags);
tcpNode.Nodes.Add("Window Size: " +
tcpHeader.WindowSize);
tcpNode.Nodes.Add("Checksum: " +
tcpHeader.Checksum);
if (tcpHeader.UrgentPointer != "")
tcpNode.Nodes.Add("Urgent Pointer: " +
tcpHeader.UrgentPointer);
return tcpNode;
}

```

6/29/2011

Chương 13: Phân tích gói tin mạng

35

Hiện thực Sniffer mức IP

```

private TreeNode MakeUDPTreeNode(UDPHeader udpHeader)
{
    TreeNode udpNode = new TreeNode();
    udpNode.Text = "UDP";
    udpNode.Nodes.Add("Source Port: " +
udpHeader.SourcePort);
    udpNode.Nodes.Add("Destination Port: " +
udpHeader.DestinationPort);
    udpNode.Nodes.Add("Length: " + udpHeader.Length);
    udpNode.Nodes.Add("Checksum: " + udpHeader.Checksum);
    return udpNode;
}

```

6/29/2011

Chương 13: Phân tích gói tin mạng

36

Hiện thực Sniffer mức IP

```
private TreeNode MakeDNSTreeNode(byte[] byteData, int nLength)
{
    DNSHeader dnsHeader = new DNSHeader(byteData, nLength);
    TreeNode dnsNode = new TreeNode();
    dnsNode.Text = "DNS";
    dnsNode.Nodes.Add("Identification: " + dnsHeader.Identification);
    dnsNode.Nodes.Add("Flags: " + dnsHeader.Flags);
    dnsNode.Nodes.Add("Questions: " + dnsHeader.TotalQuestions);
    dnsNode.Nodes.Add("Answer RRs: " + dnsHeader.TotalAnswerRRs);
    dnsNode.Nodes.Add("Authority RRs: " + dnsHeader.TotalAuthorityRRs);
    dnsNode.Nodes.Add("Additional RRs: " +
    dnsHeader.TotalAdditionalRRs);
    return dnsNode;
}
```

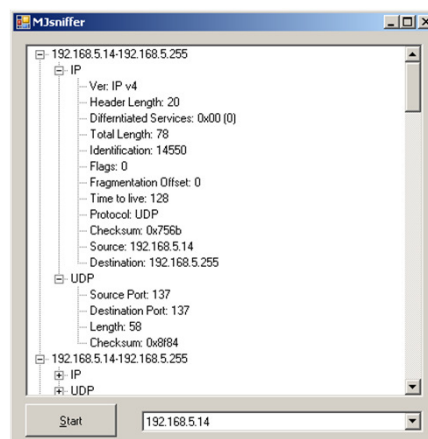
6/29/2011

Chương 13: Phân tích gói tin mạng

37

Hiện thực Sniffer mức IP

- Chạy ứng dụng, chọn card giao tiếp trong listbox, thực hiện một thao tác nào đó trên mạng, kết quả tương tự như hình



6/29/2011

Chương 13: Phân tích gói tin mạng

38

Phân tích mức Data-link

- Khi bắt gói tin ở mức 2 (Data-link), chúng ta không chỉ nhận được dữ liệu từ máy tính của mình mà còn lấy được dữ liệu từ các máy khác trong cùng mạng
- Hơn nữa, chúng ta có thể xem ARP requests, các gói NETBIOS,...
- Chúng ta có thể thử Dùng rvPacket & WinPCap (<http://www.winpcap.org>)

6/29/2011

Chương 13: Phân tích gói tin mạng

39

Phân tích mức Data-link

- WinCap DLL được thiết kế với C++ nên khó dùng trực tiếp với C#. Chúng ta có thể lựa chọn khác như PacketX (www.beesync.com)
- Ngoài ra cũng có thể dùng thư viện DLL từ rvpacket.dll (tải về từ địa chỉ <http://www.webtropy.com/downloads/rvpacket.zip>).

6/29/2011

Chương 13: Phân tích gói tin mạng

40

Dùng rvPacket & WinPCap

- Tạo project mới, gồm 1 form, 1 textbox tên tbPackets, 2 button tên btnStart, btnStop

- Xử lý sự kiện Click của button btnStop:

```
private void btnStop_Click(object sender,
EventArgs e)
{
    closeAdapter(Adapter);
}
```

6/29/2011

Chương 13: Phân tích gói tin mạng

41

Dùng rvPacket & WinPCap

- Xử lý sự kiện Click của button btnStart:

```
private void btnStart_Click(object sender, EventArgs e)
{
    short Qid;
    string packetBuffer;
    short openSuccess;
    short packetQueue;
    short packetLen;
    string rawAdapterDetails = "";
    int posDefaultAdapter;
    getAdapterNames(rawAdapterDetails);
}
```

6/29/2011

Chương 13: Phân tích gói tin mạng

42

Dùng rvPacket & WinPCap

```

Adapter = "\\"; // default adapter
openSuccess = openAdapter("\\");
if (openSuccess != ERR_SUCCESS) {
    MessageBox.Show(
        "Unable to start. Check WinPCap is installed");
    return;
}
while (true){
    packetQueue = checkPacketQueue(Adapter);
    for (Qid = 1; Qid < packetQueue; Qid++){
        packetBuffer = new

```

6/29/2011

Chương 13: Phân tích gói tin mạng

43

Dùng rvPacket & WinPCap

```

StringBuilder().Append
(' ', MAX_PACKET_SIZE).ToString();
packetLen = getQueuedPacket(packetBuffer);
packetBuffer = packetBuffer.Substring(0, packetLen);
tbPackets.Text = tbPackets.Text +
packetBuffer.Replace("\0", " ");
tbPackets.SelectionStart = tbPackets.Text.Length;
Application.DoEvents();
}
Application.DoEvents();
}
}

```

6/29/2011

Chương 13: Phân tích gói tin mạng

44

Dùng rvPacket & WinPCap

- Thư viện rvPacket được viết bằng C++ nên cần phải khai báo theo dạng Windows API.
- GetAdapterNames được dùng để lấy danh sách các card mạng, tên của những card này được chuyển cho openAdapter, hàm này mới thực sự xử lý sniffing
- CheckPacketQueue trả về số gói bắt được

6/29/2011

Chương 13: Phân tích gói tin mạng

45

Dùng rvPacket & WinPCap

- GetQueued được dùng để lấy mỗi gói tại thời điểm
- CloseAdapter thực hiện dừng quá trình sniffing và giải phóng card để cho tiến trình khác dùng được
- Khai báo các namespace:
using System.Text;
using System.Runtime.InteropServices;

6/29/2011

Chương 13: Phân tích gói tin mạng

46

Dùng rvPacket & WinPCap

- Khai báo như sau trong phần constructor của form:

```
[DllImport("rvPacket.dll")]
public static extern short getAdapterNames(string s);
[DllImport("rvPacket.dll")]
public static extern short openAdapter(string Adapter);
[DllImport("rvPacket.dll")]
public static extern short
    checkPacketQueue(string Adapter);
[DllImport("rvPacket.dll")]
public static extern short
    getQueuedPacket(string s);
```

6/29/2011

Chương 13: Phân tích gói tin mạng

47

Dùng rvPacket & WinPCap

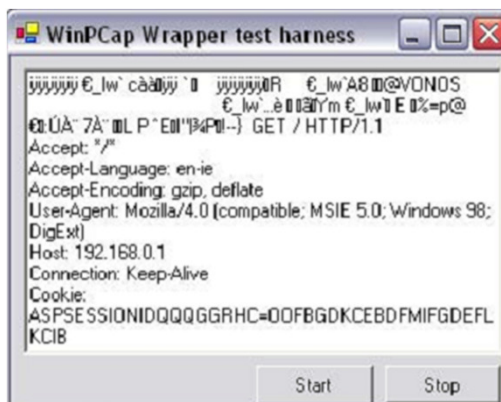
```
[DllImport("rvPacket.dll")]
public static extern void
    closeAdapter(string Adapter);
const short SIMULTANEOUS_READS = 10;
const short MAX_ADAPTER_LEN = 512;
const string ADAPTER_DELIMITER = "|";
const short MAX_PACKET_SIZE = 10000;
const short ERR_SUCCESS = 1;
const short ERR_ADAPTER_ID = 2;
const short ERR_INVALID_HANDLE = 3;
const short ERR_INVALID_ADAPTER = 4;
const short ERR_ALLOCATE_PACKET = 5;
string Adapter = "";
```

6/29/2011

Chương 13: Phân tích gói tin mạng

48

Dùng rvPacket & WinPCap



6/29/2011

Chương 13: Phân tích gói tin mạng

49

Dùng PacketX & WinPCap

- Tải control packetX tại địa chỉ www.beesync.com để minh họa
- Kiểm tra luồng TCP/IP có thể được cô lập từ luồng thô bằng cách kiểm tra 2 byte đầu tiên trong gói
- Với gói tin IP, phần IP header ngay sau frame header, tại byte thứ 14
- Byte đầu tiên trong IP header luôn là 69 khi IPv4 được dùng như chuẩn ưu tiên

6/29/2011

Chương 13: Phân tích gói tin mạng

50

Dùng PacketX & WinPCap

- Byte thứ hai trong IP header dùng để kiểm tra giao thức, và luôn là 6 nếu TCP/IP được dùng
- Chúng ta cần phải tải về và cài đặt cả hai gói WinPCap và PacketX
- Tạo project gồm 1 form, 1 textbox, 1 button
- Click phải vào Toolbox → Add tab và đặt tên cho tab mới là PacketX.

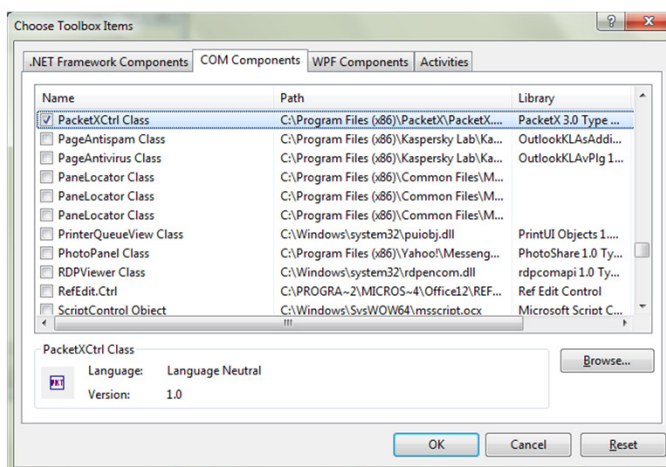
6/29/2011

Chương 13: Phân tích gói tin mạng

51

Dùng PacketX & WinPCap

- Thêm COM component mới như hình minh họa



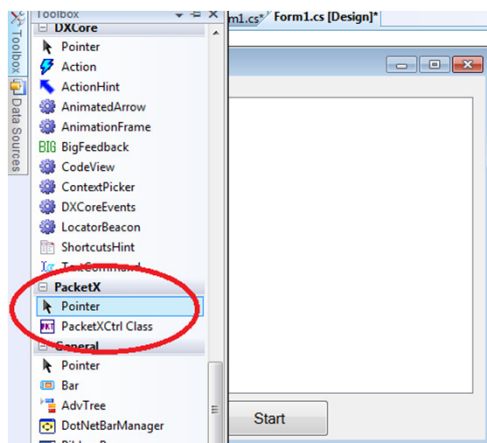
6/29/2011

Chương 13: Phân tích gói tin mạng

52

Dùng PacketX & WinPCap

- Thêm control mới vào toolbox như hình minh họa



6/29/2011

Chương 13: Phân tích gói tin mạng

53

Dùng PacketX & WinPCap

- Xử lý sự kiện Load của Form :

```
private void Form1_Load(object sender, EventArgs e)
{
    lvPackets.Columns.Add("From", lvPackets.Width / 3,
        HorizontalAlignment.Left);
    lvPackets.Columns.Add("To", lvPackets.Width / 3,
        HorizontalAlignment.Left);
    lvPackets.Columns.Add("Size", lvPackets.Width / 3,
        HorizontalAlignment.Left);
    lvPackets.View = View.Details;
}
```

6/29/2011

Chương 13: Phân tích gói tin mạng

54

Dùng PacketX & WinPCap

- Xử lý sự kiện Click của button:

```
private void btnStart_Click(object sender, EventArgs e)
{
    try
    {
        axPacketXCtrl1.Start();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

6/29/2011

Chương 13: Phân tích gói tin mạng

55

Dùng PacketX & WinPCap

- PacketX dùng một sự kiện OnPacket() để thông báo cho host biết có gói tin đến
- Nội dung của gói được lưu trữ trong một mảng byte là Data.
- Chọn đối tượng trên form, thêm phần xử lý sự kiện trên như sau:

6/29/2011

Chương 13: Phân tích gói tin mạng

56

Dùng PacketX & WinPCap

```
private void axPacketXCtrl1_OnPacket(object
sender,
AxPacketXLib._IPktXPacketXCtrlEvents_OnPacketE
vent e)
{
    string thisPacket;
    string SourceIP;
    string DestIP;
    ListViewItem item = new ListViewItem();
    thisPacket = "";
    byte[] packetData = (byte[])e.pPacket.Data;
```

6/29/2011

Chương 13: Phân tích gói tin mạng

57

Dùng PacketX & WinPCap

```
for (short l = 0; l < e.pPacket.DataSize - 1; l++){
    thisPacket = thisPacket +
    Convert.ToChar(packetData[l]);
}
if (packetData[14] == 69 && packetData[23] == 6)
{
    SourceIP = packetData[26] + "." +
    packetData[27] + "." + packetData[28] + "." +
    packetData[29];
    DestIP = packetData[30] + "." +
```

6/29/2011

Chương 13: Phân tích gói tin mạng

58

Dùng PacketX & WinPCap

```

packetData[31] + "." +
    packetData[32] + "." +
    packetData[33] + ".";
item.SubItems[0].Text = SourceIP;
item.SubItems.Add(DestIP);

item.SubItems.Add(e.pPacket.DataSize.ToString());
lvPackets.Items.Add(item);
}
}

```

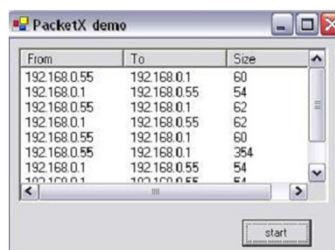
6/29/2011

Chương 13: Phân tích gói tin mạng

59

Dùng PacketX & WinPCap

- Chạy ứng dụng và chờ một kết nối TCP/IP đến hoặc có thể dùng trình duyệt mở một trang web nào đó để thực hiện bắt gói tin



6/29/2011

Chương 13: Phân tích gói tin mạng

60

Phân tích mức Physical

- Thông thường thì người ta không cần quan tâm các phần mềm đọc dữ liệu tại mức thấp, tuy nhiên ở góc độ người lập trình mạng thì chúng ta thử nghiên cứu vấn đề này
- Các thuộc tính Adapter.LinkType và Adapter.LinkSpeed của đối tượng PacketX có thể cung cấp thông tin về kiểu kết nối mạng và tốc độ truyền theo đơn vị bps

6/29/2011

Chương 13: Phân tích gói tin mạng

61

Phân tích mức Physical

- Với .NET thì class NetworkInformation cung cấp một cơ chế đơn giản để xác định máy tính có đang kết nối vào mạng không

Mã kiểu kết nối	Ý nghĩa
0	Không kết nối
1	Ethernet (802.3)
2	Token Ring (802.5)
3	FDDI (Fiber Distributed Data Interface)
4	WAN (Wide Area Network)
5	LocalTalk

6/29/2011

Chương 13: Phân tích gói tin mạng

62

Phân tích mức Physical

Mã kiểu kết nối	Ý nghĩa
6	DIX (DEC- Intel - Xerox)
7	ARCNET (raw)
8	ARCNET (878.2)
9	ATM (Asynchronous Transfer Mode)
10	Wireless

- Sử dụng cũng tương đối đơn giản như minh họa:

```
NetworkInformation netInfo = new NetworkInformation();
If (netInfo.GetIsConnected() == true)
{
    // xử lý với tình huống đã kết nối vào mạng
}
```

6/29/2011

Chương 13: Phân tích gói tin mạng

63

Phân tích mức Physical

- Class NetworkInformation kế thừa từ System.Net.NetworkInformation có nhiều thuộc tính có ích, mô tả các hoạt động ở mức thấp
- Class ActiveUdpListener được trả về sau khi dùng GetActiveUdpListeners – tương đương với gọi hàm GetUdpTable trong Windows API, hoặc thực hiện dòng lệnh NETSTAT -p udp -a

6/29/2011

Chương 13: Phân tích gói tin mạng

64

Class NetworkInformation

Phương thức hoặc thuộc tính	Mục đích
AddressChanged	Thiết lập AddressChangedEventHandler (Object,EventArgs) delegate
GetActiveUdpListeners	Liệt kê tất cả các port UDP đang hoạt động
GetIcmpV4Statistics	Trích chọn thống kê của hoạt động ping (ICMP). Trả về IcmpV4Statistics
GetIPStatistics	Trích chọn thống kê của hoạt động IP. Trả về IPStatistics
GetIsConnected	Xác định máy tính có nối vào mạng không. Trả về Boolean
GetNetworkInterfaces	Trích chọn thông tin về phần cứng mạng đã kết nối. Trả về NetworkInterface[].

6/29/2011

Chương 13: Phân tích gói tin mạng

65

Class NetworkInformation

Phương thức hoặc thuộc tính	Mục đích
GetTcpConnections	Trích chọn thống kê của hoạt động TCP/IP. Trả về TcpStatistics
GetUdpStatistics	Trích chọn thống kê của hoạt động UDP/IP. Trả về UdpStatistics
DhcpScopeName	Lấy tên phạm vi DHCP scope name. Trả về String
DomainName	Lấy tên miền đã đăng ký cục bộ. Trả về String
HostName	Lấy tên máy tính cục bộ. Trả về String
IsWinsProxy	Xác định có phải máy tính đang hoạt động như một WINS proxy. Trả về Boolean.
NodeType	Lấy kiểu NetBIOS node của máy tính. Trả về NodeType (ví dụ: broadcast, P2P, mixed, hybrid)

6/29/2011

Chương 13: Phân tích gói tin mạng

66

Một số class quan trọng khác

- Class `ActiveUdpListener`: thuộc tính `LocalEndPoint` cho biết vị trí logic của port giữ kết nối UDP đang hoạt động. Trả về `EndPoint`
- Một số class quan trọng khác như `IcmpV4Statistics`, `IPStatistics`, `NetworkInterface`, `InterfaceStatistics`, `IPAddressInformation`, `IPv4Properties`, `TcpStatistics`, `UdpStatistics` xem thêm ở tài liệu tham khảo

6/29/2011

Chương 13: Phân tích gói tin mạng

67

Bài tập

- Cài đặt các chương trình đã minh họa trong bài giảng của chương bằng ngôn ngữ C# hoặc VB.NET

6/29/2011

Chương 13: Phân tích gói tin mạng

68