实 验 十二　网络抓包工具 SharpPCap

12.1　实验目的

多种网络协议规定计算机各自目的下的通信，网络数据在网络设备上是透明及广播的传输方式，极易被截获甚至被伪造，网络通信不够稳定更受到各种攻击的威胁。本实验介绍采用 SharpPCap 工具包进行网络数据抓包的方法，加深对网络安全威胁的理解。

12.2　开发工具与网络协议

12.2.1　抓包工具 SharpPCap

网络抓包又叫包嗅探，它能够获取经过指定网卡的所有数据包，进行网络监探包分析和安全工具任务，比较出名的网络嗅探软件有 Sniffer 和 WireShark。包嗅探程序开发包有 UNIX 平台的 libpcap 库和 Windows 平台的 WinPcap 库，它们提供开发低层网络监听功能的 API。

SharpPcap 是 SourceForge 的一个开源项目，它提供了.NET 平台下的基于 libpcat/WinPcap 库的开发包，其中 API 包括网络包捕获、注入、分析包构造功能。用户可到 sourceforge.net 网站下载 SharpPcap 项目文件，它包括了必要的程序开发包和示例的源代码文件。SharpPcap 基于 WinPcap 和 AirPcap 驱动，winpcap 是 windows 平台包捕获驱动，AirPcap 专门用于 Windows 平台的的无线网络包捕获。SharpPcap 支持解析和构造数据包的网络协议有 Ethernet、IPv4 和 IPv6、Tcp、Udp、ARP、ICMP、PPPoE 等，它还能指定条件对数据包进行过滤。

使用 SharpPcap 开发包捕获项目需要引用开发库 PacketDotNet.dll 和 SharpPcap.dll 文件，SharpPcap 负责调用 libpcap/winpcap 驱动，要安装 WinPcap 驱动包，PacketDotNet 实现数据包解析与构造，在源程序中则要添加命名空间 PacketDotNet 和 SharpPcap。

12.2.2　地址解析协议 ARP

计算机通信软件使用 TCP/IP 协议进行通信用 IP 地址标识网络端点，以太网设备比如网卡都有自己全球唯一的 MAC 地址，在局域网内是以 MAC 地址来传输以太网数据包的，在以太网中进行 IP 通信就需要一个协议来建立 IP 地址与 MAC 地址的对应关系，IP 数据包才能发送出去，这个协议就是 ARP(Address Resolution Protocol，地址解析协议) 协议。ARP 将网络层地址解析为数据链路层的物理地址。

计算机 A 要和目标机器 B 通信的时候，首先检查 arp 缓存，查找是否有对应的 arp 条目，如果没有，计算机 A 广播一个 ARP 请求报文（携带主机 A 的 IP 地址 Ia——物理地址 Pa），请求 IP 地址为 Ib 的主机 B 回答物理地址 Pb，网络中每台主机都收到这个请求包，只有主机

B 识别自己的 IP 地址，其它机器均忽略请求包。主机 B 向 A 主机发回一个 ARP 响应报文，其中就包含有 B 的 MAC 地址，A 接收到 B 的应答后，就会更新本地的 ARP 缓存，有后续的数据通信中机器 A 可在通信包中包含机器 B 的 MAC 地址。

ARP 协议是早期的网络协议，采用信任模式，没有考虑网络安全。在局域网中，黑客接收到 ARP Request 广播包，能够监听到其它节点的 (IP, MAC) 地址, 黑客能够伪装为 B，告诉 A (受害者) 一个假地址，使得 A 在发送给 B 的数据包都被黑客截取，而 A, B 对数据被截获一无所知，ARP 地址欺骗经常引起网络不通现象。

12.3 实验内容

本小节任务包含两个项目，项目 A 负责获取远程主机 MAC 地址，并发送 WoL 网络包，项目 B 负责捕获项目 A 发出的网络数据。

12.3.1 获取局域网内它机的 MAC 地址

Windows 平台的库文件 Iphlpapi.dll 中提供 SendARP 函数，该函数能方便地获取远程主机 MAC 地址。函数 RemoteIpToMac 能够获取局域网内远程主机的 MAC 地址，远程主机以 IP 地址标识，通过调用 SendArp 函数收到以整数表示的 MAC 地址，需指定整数是以 16 为基将其转化为字符串。参考代码如下：

```
[DllImport("Iphlpapi.dll")]
private static extern int SendARP(
  Int32 dest,Int32 host,ref Int64 mac,ref Int32 length);
[DllImport("Ws2_32.dll")]
private static extern Int32 inet_addr(string ip);
private string RemoteIpToMac(string destIp)
{
  string temp1="",temp2="";
  try
  {
    StringBuilder macAddress = new StringBuilder();
    Int32 remote = inet_addr(destIp);
    Int64 macInfo = new Int64();
    Int32 length = 6;
    SendARP(remote, 0, ref macInfo, ref length);
    if (length == 0)
    {
      temp2 = "";
    }
    else
    {
      //两个字符代表一个字节
```

```
            temp1 = Convert.ToString(macInfo, 16).PadLeft(12, '0').ToUpper();
            for (int i = 0; i < 6; i++)
            {
                temp2 = temp2 + temp1.Substring(10 - i * 2, 2);
            }
        }

    }catch(Exception err)
    {
        MessageBox.Show("IpToMac" + err.Message);
    }
    return temp2;
}
```

12.3.2 截获 WoL 数据包

WoL 协议称为远程唤醒，它能够网络启动支持 WoL 协议的远程主机。使用 WoL 协议进行远程开机有两个步骤：1. 确定远程主机 IP 地址，在远程主机开机状态下利用 ARP 协议获取远程主机 MAC 地址；2. 在远程主机关机状态下，其它机器使用远程主机的 MAC 地址构造 UDP 数据包，广播 UDP 数据包实现远程开机。经过上小节获取到局域网内它机 MAC 地址，可构造 WoL 协议要用到的 MAC 地址。发送 WoL 数据包的参考代码如下：

```
private void sendWol(string destMac)
{
    try
    {
        byte[] macAddr = new byte[6];
        for (int i = 0; i < 6; i++)
        {
            macAddr[i] = (byte)int.Parse(destMac.Substring(i * 2, 2), System.Globalization.NumberStyles.HexNum
        }
        Socket socket_send;
        //创建一个进行 UDP 广播的 Socket 对象
        socket_send = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);
        socket_send.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.Broadcast,
1);
        IPEndPoint RemoteIpEndPoint = new IPEndPoint(IPAddress.Broadcast, 9095);
        byte[] send_data_buf = new byte[1024];
        byte[] b_txt1 = { 0xff, 0xff, 0xff, 0xff, 0xff, 0xff };
        Buffer.BlockCopy(b_txt1, 0, send_data_buf, 0, 6);
        for (int i = 1; i <= 16; i++)
```

```
        {
            Buffer.BlockCopy(macAddr, 0, send_data_buf, 6 * i, 6);
        }
        int send_data_len = 6 * 17;
        socket_send.SendTo(send_data_buf, send_data_len, SocketFlags.None, RemoteIpEnd-
Point);
        socket_send.Close();
      }
      catch (Exception err)
      {
        MessageBox.Show(err.Message);
      }
    }
```

新建一个窗体应用程序作为项目 B，添加库引用 PacketDotNet 和 SharpPcap，添加有关窗体自定义消息处理函数：

```
    [DllImport("User32.dll")]
    private static extern int SendMessage(
      IntPtr hWnd,
      int Msg,
      int wParam,
      int lParam
      );
    public const int UPDATE_CAP=0x500;
    public static IntPtr mainWndHandle;
    protected override void DefWndProc(ref Message m)
    {
      switch(m.Msg)
      {
        case UPDATE_CAP:
          textBox1.AppendText(strCap+"\r\n");
          break;
        default:
          base.DefWndProc(ref m);
          break;
      }

    }
    private static string strCap;
```

网卡捕获到网络数据包的回去调处理函数代码：

```
    private static void device_OnPacketArrival(object sender, CaptureEventArgs e)
```

```
{
    var time = e.Packet.Timeval.Date;
    var len = e.Packet.Data.Length;
    strCap=string.Format("{0}:{1}:{2},{3} Len={4}",
        time.Hour, time.Minute, time.Second, time.Millisecond, len);
    SendMessage(mainWndHandle, UPDATE_CAP, 100, 100);
    // parse the incoming packet
    var packet = PacketDotNet.Packet.ParsePacket(e.Packet.LinkLayerType, e.Packet.Data);
    if (packet == null)
        return;
    var wol = PacketDotNet.WakeOnLanPacket.GetEncapsulated(packet);
    //if (wol.PayloadData != null)
    if (wol != null)
    {
        byte[]macAddB = wol.DestinationMAC.GetAddressBytes();
        strCap = string.Format("{0:X2}-{1:X2}-{2:X2}-{3:X2}-{4:X2}-{5:X2}", macAddB[0], macAddB[1],
macAddB[2], macAddB[3], macAddB[4], macAddB[5]);
        SendMessage(mainWndHandle, UPDATE_CAP, 100, 100);
    }
}
```

为了不影响窗体响应，抓包任务由工作线程完成：

```
public static ICaptureDevice device;
private static void capData()
{
    device.OnPacketArrival +=
        new PacketArrivalEventHandler(device_OnPacketArrival);
    int readTimeoutMilliseconds = 1000;
    device.Open(DeviceMode.Promiscuous, readTimeoutMilliseconds);
    device.Filter = "ether dst FF:FF:FF:FF:FF:FF and udp";
    // start capture packets
    device.Capture();
    device.Close();
}
```

本机网卡查找与启动抓包线程：

```
private void button1_Click(object sender, EventArgs e)
{
    string ver = SharpPcap.Version.VersionString;
    CaptureDeviceList devices = CaptureDeviceList.Instance;
    if (devices.Count < 1)
    {
```

```
        textBox1.AppendText(" 没有发现网卡 \r\n");
        textBox1.ScrollToCaret();
        return;
    }
    foreach (var dev in devices)
    {
        textBox1.AppendText(string.Format("{0}\r\n",dev.Description));
    }
    device = devices[0];
    ThreadStart theSta = new ThreadStart(capData);
    Thread thr = new Thread(theSta);
    thr.Start();
}
```

在同一局域网内先启动 B 机上项目 B 的抓包任务，再启动 A 机的项目 A，A 机将发出 WoL 网络数据，B 机将捕获到该网络包。

12.4 实验作业

1. 完成本实验中的程序项目。
2. 尝试编写具有 ARP 地址欺骗功能的函数。