

[TOOL] ARP Promiscuous Node Detection

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2003-01/0010.html>

From: support@securiteam.com

Date: 01/05/03

From: support@securiteam.com

To: list@securiteam.com

Date: 5 Jan 2003 11:31:46 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

Beyond Security would like to welcome Tiscali World Online to our service provider team.

For more info on their service offering IP-Secure, please visit http://www.worldonline.co.za/services/work_ip.asp

ARP Promiscuous Node Detection

DETAILS

APD is a promiscuous node detection tool which uses ARP packets to determine whether or not a host is in promiscuous mode.

Tool source code:

```
/*
 * APD v1.1b : Arp Promiscuous Node Detection
 *
 * Copyright (C) 2002 Dr.Tek of Malloc() Security
 * Contact: Dr.Tek <tek@superw00t.com>
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
```

Securiteam: [TOOL] ARP Promiscuous Node Detection

* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place – Suite 330, Boston, MA 02111–1307,
USA.
*
*
* Malloc() Security (www.malloc.tk)
*/

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <fcntl.h>
#include <net/if.h>
#include <sys/ioctl.h>
#include <sys/signal.h>
#include <netdb.h>
#include <string.h>

#if __BYTE_ORDER == __BIG_ENDIAN
#define HFIX_S(x) (x)
#define HFIX_L(x) (x)
#define NFIX_S(x) (x)
#define NFIX_L(x) (x)
#elif __BYTE_ORDER == __LITTLE_ENDIAN
#define HFIX_S(x) htons(x)
#define HFIX_L(x) htonl(x)
#define NFIX_S(x) ntohs(x)
#define NFIX_L(x) ntohl(x)
#endif

#ifndef ETH_P_ARP
#define ETH_P_ARP 0x0806
#endif

#ifndef SIOCGIFCONF
#define SIOCGIFCONF 0x8912
#endif

#define SIOCGIFHWADDR 0x8927

#ifndef TIMEOUT
#define TIMEOUT 100
#endif

typedef unsigned char uchar;
typedef unsigned short u16;
typedef unsigned long u32;
```

Securiteam: [TOOL] ARP Promiscuous Node Detection

```
#define ARP_ETHER 1
#define ETH_HWLEN 6
#define ARP_FRAME 0x8060
#define IP_PTYPE 0x0800
#define OP_ARPREQ 2
#define IP_ALEN 4
#define MAX_DLEN 5

#define ARP_REQ 1 /* ARP Request opcode */
#define ARP_REP 2 /* ARP Reply opcode */

#define ETH_LEN 0x0e
#define ARP_LEN 0x1c

char*
inet_ntoa(u32);

u32
inet_addr(const char *);

/* structure for ETH Packet */
struct eth_pkt {

    uchar t_addr[ETH_HWLEN];
    uchar s_addr[ETH_HWLEN];
    u16 frame_type;
};

/* structure for ARP Packet */
struct arp_pkt {

    u16 hw_type;
    u16 prot_type;
    uchar hwaddr_sz;
    uchar praddr_sz;
    u16 op;
    uchar sndr_hw_addr[ETH_HWLEN];
    uchar sndr_ip_addr[IP_ALEN];
    uchar rcpt_hw_addr[ETH_HWLEN];
    uchar rcpt_ip_addr[IP_ALEN];
};

void apd_start_testing(u32 src, u32 dst,uchar *dev)
{
    int haddr_sckt=0 , out_sckt=0 , c=0 , sz=0 , i ,
arp_proto=HFIX_S(ETH_P_ARP);
    register int in_sckt;
    struct ifreq ifr;
    struct sockaddr sndarp;
    uchar pkt[ETH_LEN+ARP_LEN+1];
    struct eth_pkt eth_p;
```

Securiteam: [TOOL] ARP Promiscuous Node Detection

```
struct arp_pkt arp_p;
struct arp_pkt *in_a;
    uchar in_hwaddr[6];
uchar in_buf[ETH_LEN+ARP_LEN+20];
uchar pseudo_hw_d[ETH_HWLEN] = {0xFF,0xFF,0xFF,0xFF,0xFF,0xFE};
uchar pseudo_hw_s[ETH_HWLEN] = {00,11,22,33,44,55};
uchar pseudo_hw_a[ETH_HWLEN] = {0x00,0x00,0x00,0x00,0x00,0x00};

/* Retrieving Hardware Address from NIC */
strcpy(ifr.ifr_name,dev);
if((haddr_sckt=socket(AF_INET,SOCK_DGRAM,0))<0)
{
    fprintf(stderr,"Error creating pseudo socket!\n");
    exit(-1);
}

if((ioctl(haddr_sckt,SIOCGIFHWADDR,&ifr)<0)
{
    perror("ioctl:");
    close(haddr_sckt);
    exit(-1);
}
memcpy(in_hwaddr,(uchar *)ifr.ifr_hwaddr.sa_data,6);
close(haddr_sckt);

printf("==> Probing host: %s\n",inet_ntoa(dst));
sleep(2);

/* Creating out socket! */
if((out_sckt=socket(AF_INET,SOCK_PACKET, arp_proto ))<0)
{
    fprintf(stderr,"Error creating socket!\n");
    exit(-1);
}

/* Creating in socket! */
if((in_sckt=socket(AF_INET,SOCK_PACKET, arp_proto))<0)
{
    fprintf(stderr,"Error creating socket!\n");
    exit(-1);
}

bzero(pkt,ETH_LEN+ARP_LEN);

memcpy(&eth_p.t_addr,pseudo_hw_d,ETH_HWLEN);
memcpy(&eth_p.s_addr,pseudo_hw_s,ETH_HWLEN);
eth_p.frame_type = HFIX_S(ETH_P_ARP);
memcpy(pkt,&eth_p,ETH_LEN);
```

Securiteam: [TOOL] ARP Promiscuous Node Detection

```
arp_p.hw_type = HFIX_S(ARP_ETHER);
arp_p.prot_type = HFIX_S(IP_PTYPE);
arp_p.hwaddr_sz = ETH_HWLEN;
arp_p.praddr_sz = IP_ALEN;
arp_p.op = HFIX_S(ARP_REQ);

memcpy(&arp_p.sndr_hw_addr, in_hwaddr, ETH_HWLEN);
memcpy(&arp_p.sndr_ip_addr,&src,4);
memcpy(&arp_p.rcpt_hw_addr,pseudo_hw_a ,ETH_HWLEN);
memcpy(&arp_p.rcpt_ip_addr, &dst,4);
memcpy(pkt+ETH_LEN,&arp_p,ARP_LEN);

strcpy(sndarp.sa_data,dev);
if((sendto(out_sckt,pkt,ETH_LEN+ARP_LEN,0,&sndarp,sizeof(sndarp)))<0)
{
    fprintf(stderr,"Error sending packet!\n");
    exit(-1);
}

close(out_sckt);

i = fcntl(in_sckt,F_GETFL);
if ((fcntl(in_sckt,F_SETFL,i | O_NONBLOCK)<0)
{
    perror("fcntl:");
    close(in_sckt);
    exit(-1);
}

/* Awaiting ARP Reply from possible culprit! */
while(c<TIMEOUT)
{
    bzero(in_buf,ETH_LEN+ARP_LEN);
    if((sz=read(in_sckt,in_buf,ETH_LEN+ARP_LEN))<0)
    {
        usleep(50000);
        c++; /* Wait, isn't this a programming language
:P */
    }

    if(sz>=ETH_LEN+ARP_LEN)
    {
        in_a = (struct arp_pkt *) (in_buf+ETH_LEN);
        if((NFIX_S(in_a->op)) == ARP_REP)
        {
            fprintf(stdout,"==> %s is in promiscuous
mode.\n\n",inet_ntoa(dst)); fflush(stdout);
            close(in_sckt);
        }
    }
}
```

Securiteam: [TOOL] ARP Promiscuous Node Detection

```
        sleep(2);
        break;
    }

}

    if(c==TIMEOUT)
        {
            fprintf(stdout,"==> %s is not in
promiscuous mode.\n\n",inet_ntoa(dst));
            close(in_sckt);
            sleep(2);
        }

}
void apd_banner()
{

    printf("\n\nAPD v1.1b : ARP Promiscuous Node Detection.\n");
    printf("Written by: Dr.Tek of Malloc() Security\n\n");
}

u32 get_our_addr()
{
    u32 i;
    char hname[80+1];
    struct hostent *h;

    gethostname(hname,80);

    i=inet_addr(hname);
    if(i == -1)
    {
        if((h=gethostbyname(hname))==NULL)
        {
            fprintf(stderr,"Invalid hostname.\n");
            exit(-1);
        }
        bcopy(h->h_addr,(char *)&i,h->h_length);
    }

    return i;

}

u32 resv_addr(char *haddr)
{
    u32 i;
```

Securiteam: [TOOL] ARP Promiscuous Node Detection

```
struct hostent *h;

i=inet_addr(haddr);
if(i == -1)
{

    if((h=gethostbyname(haddr))==NULL)
    {
        fprintf(stderr,"Invalid host\n");
        exit(-1);
    }

    bcopy(h->h_addr,(char *)&i,h->h_length);
}

return i;
}

void sig_handler(int sig);

int main(int argc , char **argv)
{
    int o;
    u32 ip_s ,ip_e, ip_t , tmp;
    char iface_dev[MAX_DLEN];
    iface_dev[MAX_DLEN]='\0';
    bzero(iface_dev,MAX_DLEN);
    apd_banner();
    if(argc<6)
    {
        fprintf(stderr,"%s [options]\n\n",argv[0]);
        fprintf(stderr,"Options:\n\n");
        fprintf(stderr,"-s addr : Start address.\n");
        fprintf(stderr,"-e addr : End address.\n");
        fprintf(stderr,"-d dev : network device.\n\n");
        exit(0);
    }

    while((o = getopt(argc ,argv ,"s:e:d:"))!=EOF)
    {

        switch(o) {

            case 's':
                ip_s=resv_addr((char *)optarg);
                break;

            case 'e':
                ip_e=resv_addr((char *)optarg);
```

Securiteam: [TOOL] ARP Promiscuous Node Detection

```
break;

    case 'd':
    strncpy(iface_dev,(char *)optarg,MAX_DLEN-1);
    break;

    }

}

ip_t=get_our_addr();

signal(SIGINT ,sig_handler);
signal(SIGTERM,sig_handler);
signal(SIGHUP ,SIG_IGN);

printf("-----[ APD starting ]-----\n\n");
sleep(2);
apd_start_testing(ip_t,ip_s,iface_dev);
ip_s = NFIX_L(ip_s);
tmp = HFIX_L(++ip_s);

while(tmp<=ip_e)
{
    apd_start_testing(ip_t,tmp,iface_dev);
    tmp = HFIX_L(++ip_s);
    sleep(2);
    if(tmp<iip_s) break;
}

printf("-----[ APD ending ] -----\n\n");
exit(0);
}

void sig_handler(int sig)
{

    printf("Exiting!\n\n");
    exit(0);
    sleep(2);
}


```

ADDITIONAL INFORMATION

The information has been provided by <mailto:tek@superw00t.com> Dr. Tek.

=====

This bulletin is sent to members of the SecuriTeam mailing list.
To unsubscribe from the list, send mail with an empty subject line and body to:

Securiteam: [TOOL] ARP Promiscuous Node Detection

list-unsubscribe@securiteam.com

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====
=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.